ภาษาซีและ Arduino

ฟรี ! โครงสร้างชุดทดลองการควบคุมตำแหน่งแบบ PID

ARDUINO

DEV C++

PID

Robot

C & Arduino UNO

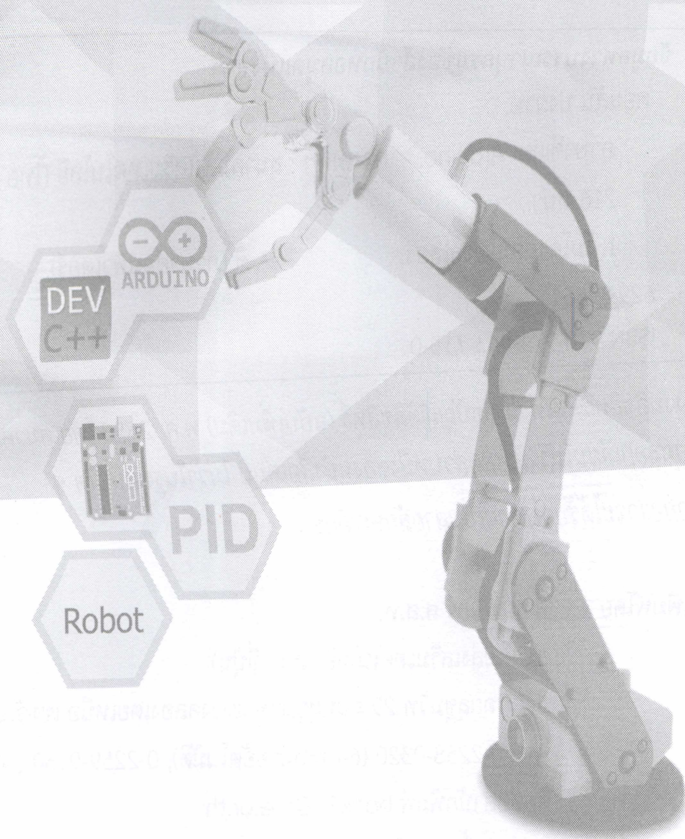
อ่านง่าย เข้าใจ ประยุกต์ใช้ได้จริง



สำนักพิมพ์ ส.ส.ท.
สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)

ผู้ช่วยศาสตราจารย์ดอนสัน ปงผาบ

ภาษาซีและ Arduino



โดย

ผู้ช่วยศาสตราจารย์ดอนสัน ปงผาบ



สำนักพิมพ์ ส.ส.ท.
สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)

295.-

ภาษาซีและ Arduino



โดย... **ผู้ช่วยศาสตราจารย์ดอนสัน ปงผาบ**

ราคา 295 บาท

พิมพ์ครั้งที่ 1-4 กันยายน 2560 - ตุลาคม 2561
พิมพ์ครั้งที่ 5 กรกฎาคม 2562
พิมพ์ครั้งที่ 6 มิถุนายน 2564

ข้อมูลทางบรรณานุกรมของสำนักหอสมุดแห่งชาติ
ดอนสัน ปงผาบ.
ภาษาซีและ Arduino. -- กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2560.
216 หน้า.
1. ไมโครคอนโทรลเลอร์. 2. ซี (ภาษาคอมพิวเตอร์). I. ชื่อเรื่อง.
629.89551
ISBN 978-974-443-715-0

จัดพิมพ์โดย **สำนักพิมพ์ ส.ส.ท.**
สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
5-7 ซอยสุขุมวิท 29 ถนนสุขุมวิท แขวงคลองเตยเหนือ เขตวัฒนา กรุงเทพฯ 10110
โทร. 0-2258-0320 (6 เลขหมายอัตโนมัติ), 0-2259-9160 (10 เลขหมายอัตโนมัติ)
ติดต่อสำนักพิมพ์ book4u@tpa.or.th
ติดต่อสั่งซื้อหนังสือ www.tpabook.com

จัดจำหน่ายโดย **บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน)**
เลขที่ 1858/87-90 ถนนบางนา-ตราด
แขวงบางนา เขตบางนา กรุงเทพฯ 10260
โทร. 0-2739-8000, 0-2739-8222 โทรสาร 0-2739-8356-9
www.se-ed.com





“ถ้าหนังสือมีข้อผิดพลาดเนื่องจากการพิมพ์ ให้นำมาแลกเปลี่ยนได้ที่สมาคมฯ” โทร. 0-2258-0320 ต่อ 1560, 1570

■ **บรรณาธิการบริหาร** สุกัญญา จารุการ **บรรณาธิการเล่ม** พรรณพิมล กิจไพฑูรย์ **กองบรรณาธิการ** รินดา คันธวร, วัลภา ลีรัชดานนท์, แสงเงิน นาคพัฒน์ **ออกแบบปก** ธารินี คุตตะสิงคี **ออกแบบรูปเล่ม** ประเทือง คชเสนีย์ **ศิลปกรรม** ศิรินุช อิศรางกูร ณ อยุธยา **ธุรการสำนักพิมพ์** อังคณา อรรณพงศ์ธร ■ **พิมพ์ที่** : ห้างหุ้นส่วนจำกัด ที.เอส.บี. โปรดักส์

เกี่ยวกับ สำนักพิมพ์ ส.ส.ท.

สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) หรือ ส.ส.ท. ก่อตั้งอย่างเป็นทางการในวันที่ 24 มกราคม พ.ศ. 2516 จากความตั้งใจ มุ่งมั่น ความร่วมมือร่วมใจ และความเสียสละ ทุ่มเทกำลังกายและกำลังใจของกลุ่มบุคคลที่เคยไปศึกษาและดูงานโดยทุน ABK & AOTS ณ ประเทศญี่ปุ่น โดยมี ฯพณฯ สมหมาย ฮุนตระกูล เป็นประธานคณะกรรมการก่อตั้ง และสำเร็จด้วยความช่วยเหลืออย่างดีจากอาจารย์โงอิจิ โฮซูมิ อดีตประธานกรรมการสมาคมความร่วมมือทางเศรษฐกิจญี่ปุ่น-ไทย (JTECS) ทั้งนี้โดยได้รับความช่วยเหลือทางด้านเงินทุนจากกระทรวงการค้าระหว่างประเทศและอุตสาหกรรม (METI) ประเทศญี่ปุ่น ซึ่งไม่มีพันธะผูกพันใด ๆ เพื่อใช้จ่ายในการดำเนินกิจกรรมต่าง ๆ

สำหรับ สำนักพิมพ์ ส.ส.ท. จัดตั้งขึ้นเมื่อ พ.ศ. 2516 พร้อม ๆ กับการก่อตั้งสมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) โดยใช้ชื่อว่า โครงการตำรา เพื่อมุ่งส่งเสริมให้มีตำราภาษาไทยเกี่ยวกับเทคโนโลยีต่าง ๆ โดยเฉพาะอย่างยิ่งในระดับอาชีวศึกษา ซึ่งในขณะนั้นยังมีอยู่จำกัดให้แพร่หลายขึ้น เพื่อยกระดับมาตรฐานการศึกษาในสายวิชาชีพซึ่งเป็นกำลังสำคัญในการพัฒนาอุตสาหกรรมไทย

ในระยะ 4-5 ปี หลังจากการก่อตั้งสมาคมฯ โครงการตำราได้เปลี่ยนชื่อเป็น โครงการสนับสนุนเทคนิคอุตสาหกรรม และได้ขยายกิจกรรมเป็น ส่วนตำราสนับสนุนเทคนิคอุตสาหกรรม ในปี พ.ศ. 2539 พร้อม ๆ กับการขยายขอบข่ายหนังสือที่จัดพิมพ์ให้ครอบคลุมหนังสือด้านการบริหารจัดการธุรกิจ อุตสาหกรรม และจิตวิทยา-การพัฒนาตนเอง รวมถึงคณิตศาสตร์ วิทยาศาสตร์ ทั้งที่เป็นงานเขียนและงานแปล ภายใต้ชื่อ สำนักพิมพ์ ส.ส.ท. โดยมีวัตถุประสงค์เพื่อถ่ายทอดและเผยแพร่ความรู้ในสาขาต่างๆ ให้แก่บุคลากรทั้งภาครัฐและเอกชน ตลอดจนนักเรียน นักศึกษา เยาวชน ซึ่งจะเป็นรากฐานสำคัญในการพัฒนาอุตสาหกรรม เศรษฐกิจ และสังคมไทยให้เติบโตได้อย่างยั่งยืนต่อไป

สำนักพิมพ์ ส.ส.ท. ใคร่ขอแสดงความขอบคุณเป็นอย่างยิ่งต่อผู้เขียนและผู้แปลทุกท่านที่มีส่วนสำคัญในความสำเร็จของสำนักพิมพ์ตั้งแต่เริ่มต้นจวบจนปัจจุบัน และหวังว่าหนังสือของสำนักพิมพ์ ส.ส.ท. จะมีส่วนช่วยในการพัฒนาบุคลากร อันจะนำไปสู่การสร้างสรรค์สังคมและเศรษฐกิจของประเทศให้ก้าวหน้าและยั่งยืน และหากท่านผู้อ่านมีข้อชี้แนะประการใด ขอได้โปรดแจ้งให้ทางสำนักพิมพ์ทราบด้วย จักเป็นพระคุณยิ่ง



สำนักพิมพ์ ส.ส.ท.

สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)

คำนำ

หนังสือ "ภาษาซีและ Arduino" ใช้สำหรับประกอบการเรียนการสอนในวิชาที่เกี่ยวกับการเขียนโปรแกรมภาษาซีในงานควบคุมไมโครคอนโทรลเลอร์ Arduino ซึ่งเป็นไมโคร-คอนโทรลเลอร์ที่ได้รับความนิยมสูงสุดในปัจจุบัน เนื้อหาของหนังสือประกอบด้วย ความรู้พื้นฐานของภาษาซี การเขียนโปรแกรมเปรียบเทียบ การเขียนโปรแกรมวนรอบ อาร์เรย์ ฟังก์ชัน พื้นฐานของ Arduino UNO ดิจิทัลอินพุตและเอาต์พุต แอนะล็อกอินพุตและเอาต์พุต การควบคุมดีซีมอเตอร์ สเต็ปเปอร์มอเตอร์ เซอร์โวมอเตอร์ ระบบควบคุมแบบ PID พื้นฐานของหุ่นยนต์และแขนกล

ผู้จัดทำได้เพียรพยายามทำให้หนังสือเล่มนี้สมบูรณ์ที่สุดอย่างเต็มกำลังความสามารถโดยเรียบเรียงตามลำดับเนื้อหา อธิบายสรุปเป็นขั้นตอน ใช้รูปภาพประกอบการอธิบาย และเสริมตัวอย่างประกอบเพื่อเพิ่มความเข้าใจ จึงหวังว่าจะเป็นประโยชน์แก่ผู้เรียนให้สามารถนำความรู้ไปประยุกต์ใช้งานได้ต่อไป ทั้งนี้หากมีข้อบกพร่องหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับคำแนะนำต่าง ๆ เพื่อใช้ในการปรับปรุงแก้ไขและพัฒนาต่อไป

สุดท้าย ผู้จัดทำขอขอบพระคุณอาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาความรู้ และทางสมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) ที่ได้สนับสนุนการจัดพิมพ์หนังสือเล่มนี้จนเสร็จสมบูรณ์

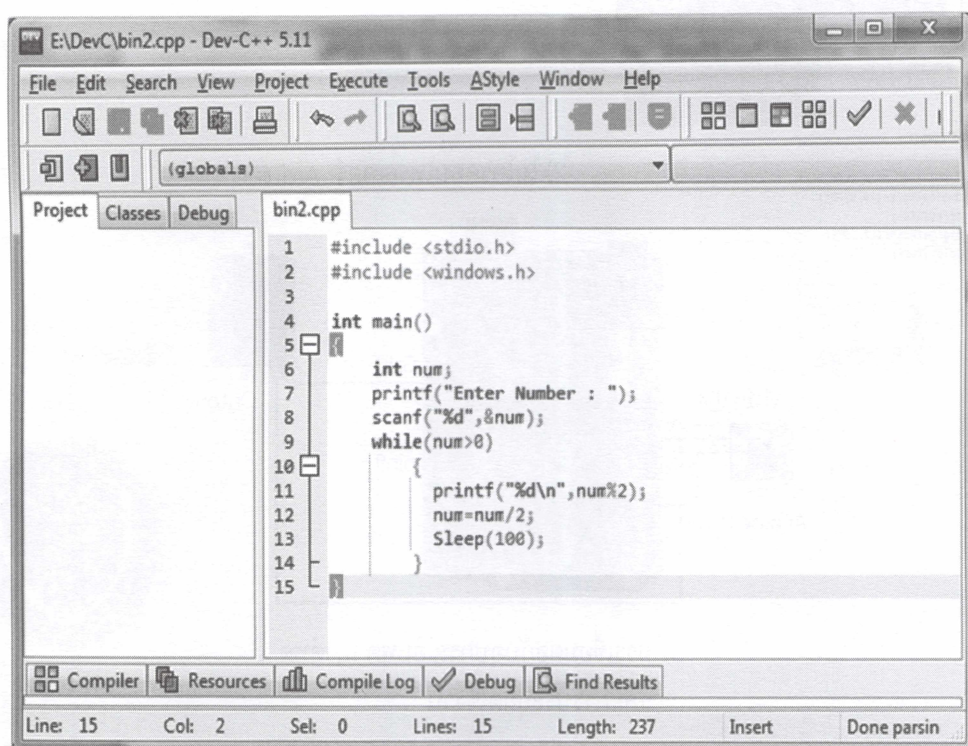
ผู้ช่วยศาสตราจารย์ดอนสัน ปงผาบ

สารบัญ



พื้นฐานภาษาซี

ภาษาซี (C Language) เป็นภาษาที่ได้รับความนิยมมาตั้งแต่อดีตจนถึงปัจจุบัน ถูกพัฒนามาจากภาษาบีโดย เดนนิส ริตชี (Dennis Ritchie) ที่ห้องปฏิบัติการเบลล์ (Bell Laboratories) ในสหรัฐอเมริการาวปี ค.ศ. 1972 เป็นภาษาที่มีความยืดหยุ่นในการเขียนโปรแกรม มีลักษณะของภาษาที่เป็นโครงสร้าง การเขียนโปรแกรมเป็นแบบลำดับ และเป็นภาษาที่ใช้คอมไพเลอร์เป็นตัวแปลความหมาย ถูกจัดกลุ่มเป็นภาษาระดับกลางของภาษาคอมพิวเตอร์ ภาษาซีสามารถประยุกต์ใช้งานในด้านต่าง ๆ ได้อย่างกว้างขวาง เช่น โปรแกรมระบบ โปรแกรมคำนวณ โดยเฉพาะไมโครคอนโทรลเลอร์ตระกูลต่าง ๆ เช่น MCS-51 PIC และ Arduino จะใช้ภาษาซีในการพัฒนาโปรแกรม ในบทที่ 1 ถึง 4 ของหนังสือเล่มนี้จะใช้โปรแกรม Dev-C++ เป็นเครื่องมือในการพัฒนาโปรแกรม



รูปที่ 1.1 โปรแกรม Dev-C++

1.1 การเขียนโปรแกรมภาษาซีในงานควบคุม

การเขียนโปรแกรมภาษาซีเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ให้สามารถประยุกต์ใช้งานได้จะต้องเรียนรู้และเข้าใจใน 4 ส่วนดังนี้

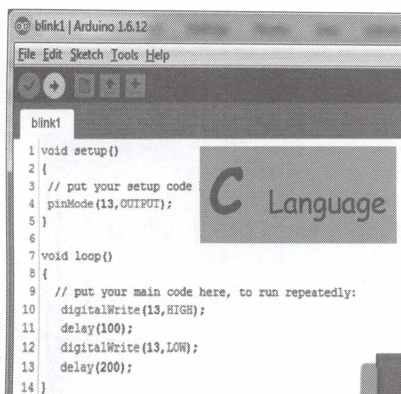
ส่วนที่ 1) ภาษาซี เรียนรู้การใช้งานโปรแกรม โครงสร้างของภาษาซี ฟังก์ชัน คำสั่ง อัลกอริทึมในการแก้ปัญหา และฝึกทักษะการเขียนโปรแกรม

ส่วนที่ 2) ไมโครคอนโทรลเลอร์ Arduino เรียนรู้โครงสร้าง ขาสัญญาณ คุณสมบัติของ Arduino ที่ใช้งาน ซึ่ง Arduino ถูกออกแบบมาให้สามารถใช้งานได้ง่าย ไม่จำเป็นต้องเข้าใจถึงโครงสร้างภายในหรือรีจิสเตอร์เหมือนไมโครคอนโทรลเลอร์รุ่นก่อน

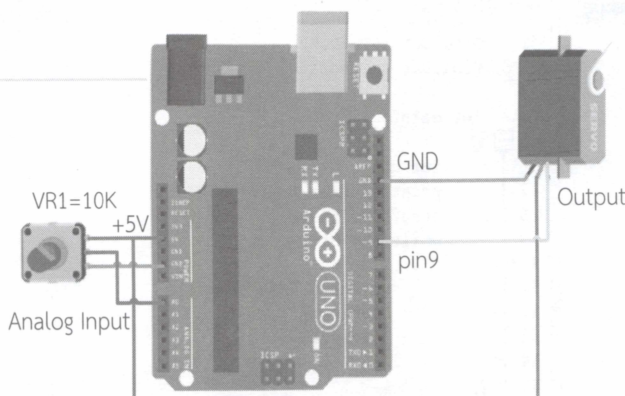
ส่วนที่ 3) การเชื่อมต่อกับอุปกรณ์อินพุต เอาต์พุต และการประยุกต์ใช้งาน ไมโครคอนโทรลเลอร์ Arduino ถูกออกแบบมาให้เชื่อมต่อกับอุปกรณ์อินพุตและเอาต์พุตได้โดยง่ายทั้งแบบดิจิทัลและแอนะล็อก มีไลบรารีฟังก์ชันให้ใช้พัฒนาระบบได้อย่างรวดเร็ว

ส่วนที่ 4) ใจรัก เพียงแค่ใจรักก็จะสำเร็จแล้วครึ่งหนึ่ง เพราะจะทำให้เราเรียนรู้ภาษาซีและไมโครคอนโทรลเลอร์อย่างมีความสุข หากขยันหมั่นฝึกฝนก็จะประสบความสำเร็จในการเรียน

1) ภาษาซี



2) ไมโครคอนโทรลเลอร์ Arduino



3) การเชื่อมต่อกับอุปกรณ์อินพุต เอาต์พุต และการประยุกต์ใช้งาน

4) ใจรัก

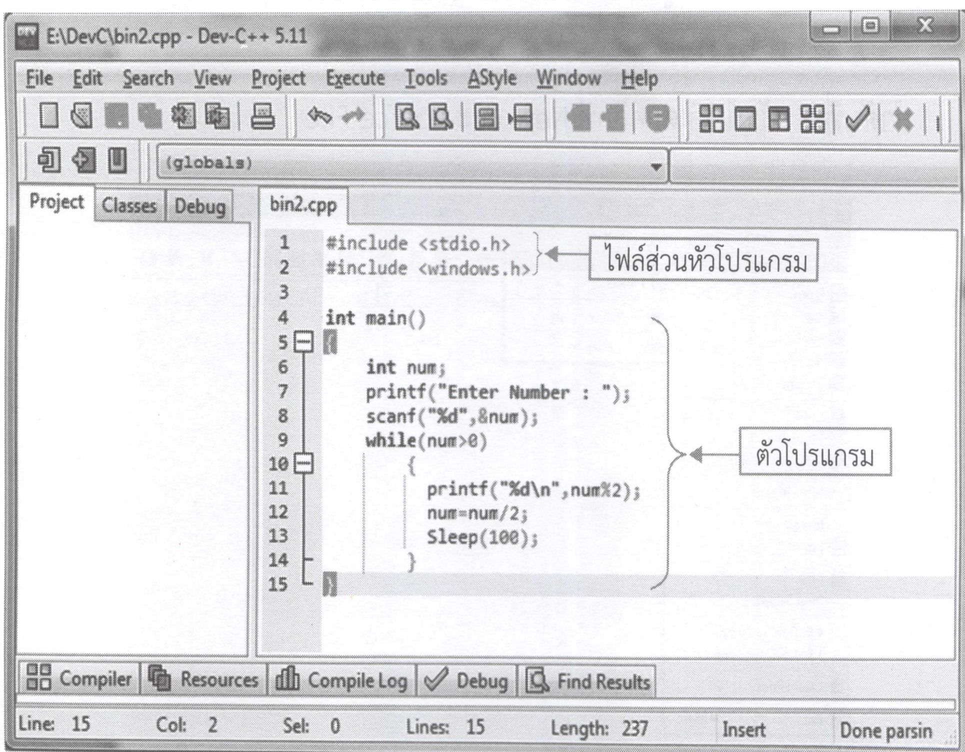
รูปที่ 1.2 ส่วนประกอบของการเขียนโปรแกรมภาษาซีในงานควบคุม

1.2 โครงสร้างของภาษาซี

โครงสร้างของภาษาซีประกอบด้วย 2 ส่วนใหญ่ ๆ คือ ไฟล์ส่วนหัวโปรแกรม (Header Files) และตัวโปรแกรม

1) **ไฟล์ส่วนหัวโปรแกรม** เป็นไฟล์ที่มีส่วนขยายเป็น *.h ใช้เก็บไลบรารีของภาษาซี เพื่อใช้ร่วมในการคอมไพล์โปรแกรม อย่างเช่น stdio.h เป็นไฟล์ที่เก็บไลบรารีมาตรฐานเกี่ยวกับการรับข้อมูลและการแสดงผล ซึ่งฟังก์ชัน printf(); ถูกนิยามไว้ใน stdio.h ดังนั้นโปรแกรมใดที่มีการเรียกใช้ฟังก์ชัน printf(); ต้องประกาศไฟล์ stdio.h เพื่อใช้ร่วมในการคอมไพล์โปรแกรมด้วย

2) **ตัวโปรแกรม** เริ่มต้นด้วยฟังก์ชัน int main() ซึ่งเป็นฟังก์ชันหลัก มีเครื่องหมายปีกกาเปิดเป็นเครื่องหมายเริ่มต้นการเขียนโปรแกรม และเครื่องหมายปีกกาปิดเป็นเครื่องหมายจบโปรแกรม ภายในฟังก์ชัน int main() ประกอบไปด้วยชุดคำสั่งและฟังก์ชันต่าง ๆ ซึ่งเกือบทั้งหมดจะปิดท้ายด้วยเครื่องหมายเซมิโคลอน การเขียนโปรแกรมภาษาซีจะแยกความแตกต่างระหว่างอักษรตัวพิมพ์เล็กและอักษรตัวพิมพ์ใหญ่ ในส่วนของคำอธิบายจะคั่นด้วยเครื่องหมาย /* และ */ ตามลำดับ หรือนำหน้าด้วยเครื่องหมาย // โดยส่วนที่เป็นคำอธิบายการทำงานของโปรแกรมจะไม่มีผลต่อการคอมไพล์

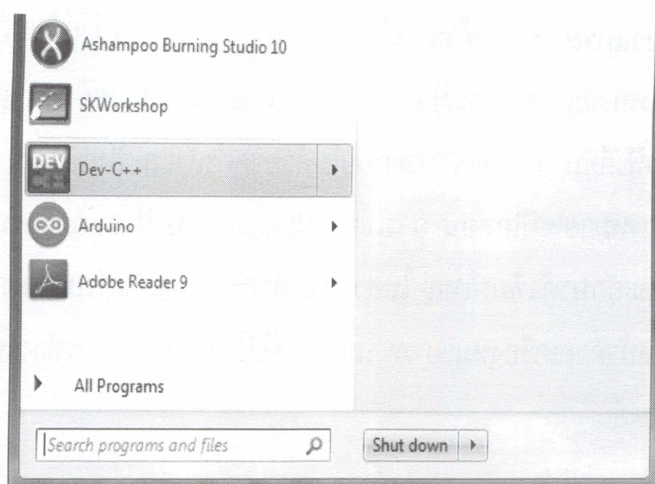


รูปที่ 1.3 ส่วนประกอบของภาษาซี

1.3 ขั้นตอนการใช้งานโปรแกรม Dev-C++

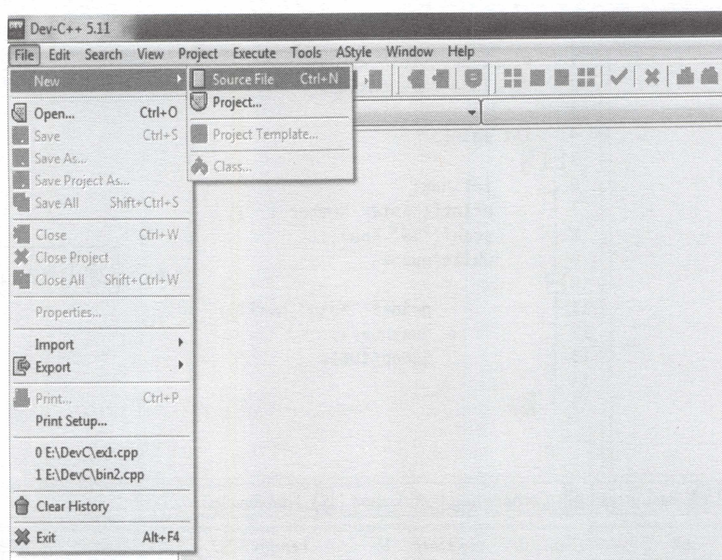
การพัฒนาโปรแกรมภาษาซีเริ่มจากเขียนโปรแกรม (Source Program) ซึ่งต้องเขียนตามลักษณะโครงสร้างของภาษาซี แล้วบันทึกไฟล์ให้มีนามสกุลเป็น *.cpp จากนั้นทำการคอมไพล์และรันโปรแกรม ถ้ามีข้อผิดพลาด ให้กลับไปแก้ไขที่โปรแกรม แต่ถ้าไม่มีข้อผิดพลาดจะได้ไฟล์ *.exe และแสดงผลของการรันโปรแกรมบนหน้าจอ ซึ่งมีขั้นตอนตามลำดับดังนี้

ลำดับที่ 1 เข้าโปรแกรม Dev-C++



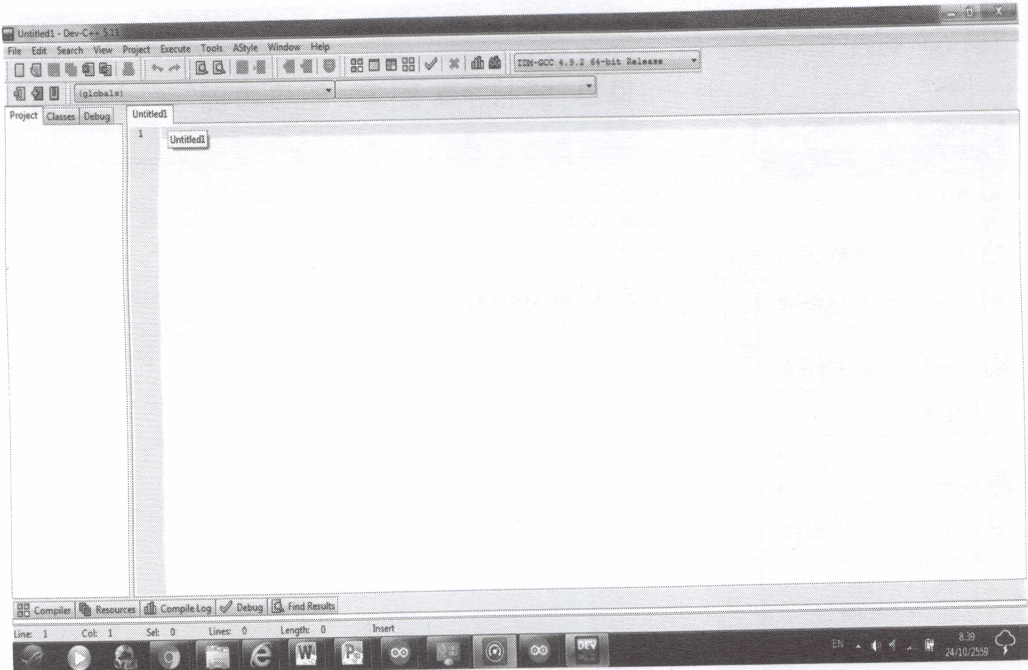
รูปที่ 1.4 การเข้าโปรแกรม Dev-C++

ลำดับที่ 2 สร้างไฟล์โดยคลิกที่ New และ Source File



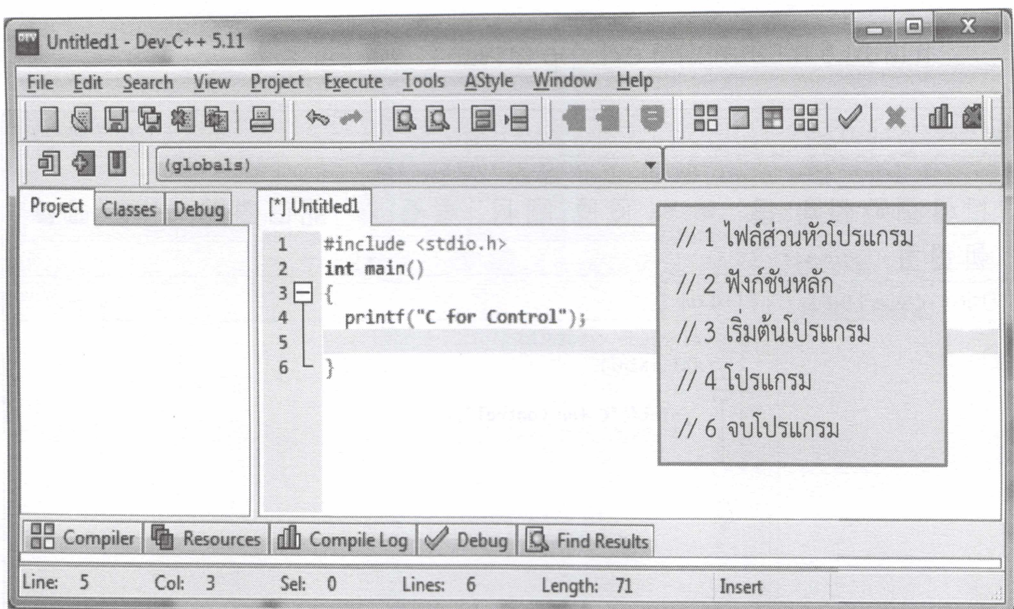
รูปที่ 1.5 การสร้างไฟล์ใหม่

ลำดับที่ 3 จะได้หน้าต่างของไฟล์ใหม่เพื่อเขียนโปรแกรม



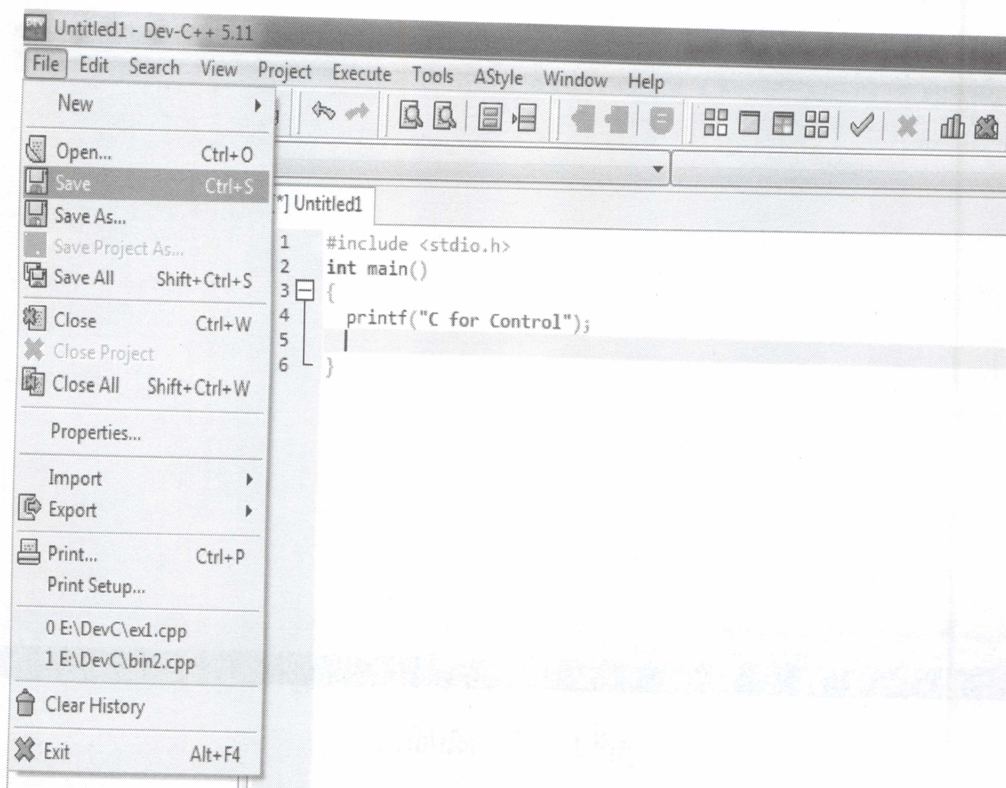
รูปที่ 1.6 หน้าต่างไฟล์ใหม่

ลำดับที่ 4 พิมพ์โปรแกรมตามตัวอย่าง



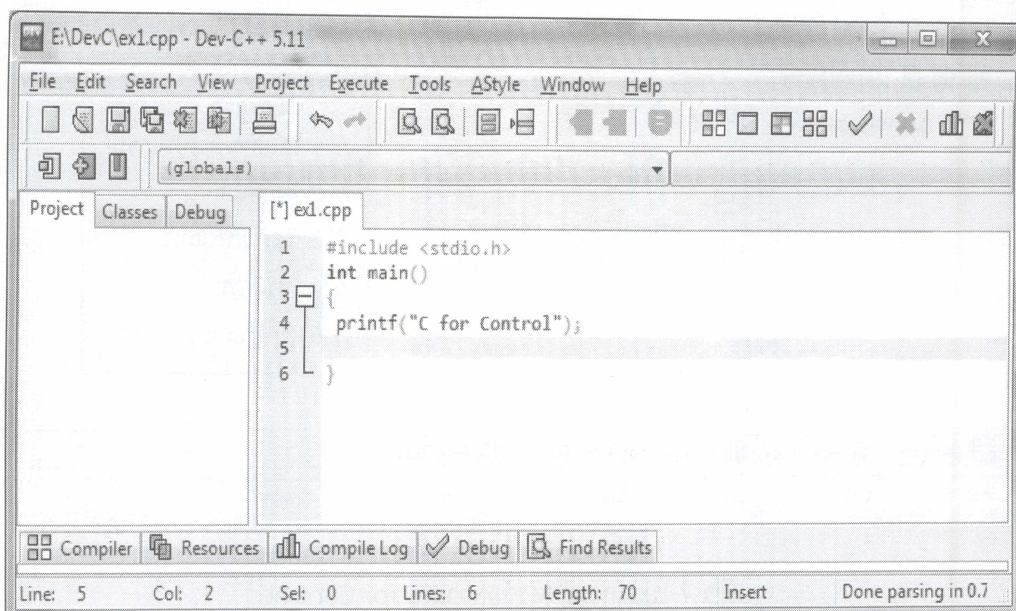
รูปที่ 1.7 โปรแกรมแสดงข้อความ C for Control

ลำดับที่ 5 ทำการบันทึกไฟล์ ex1.cpp



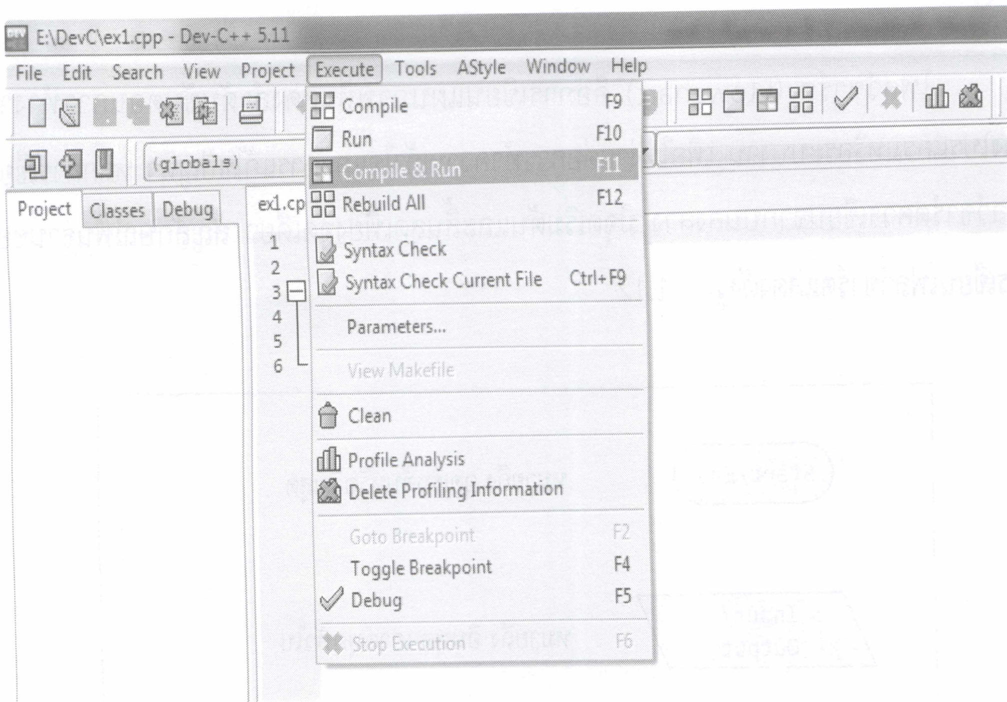
รูปที่ 1.8 การบันทึกไฟล์

ลำดับที่ 6 เมื่อบันทึกไฟล์ ex1.cpp เสร็จแล้ว



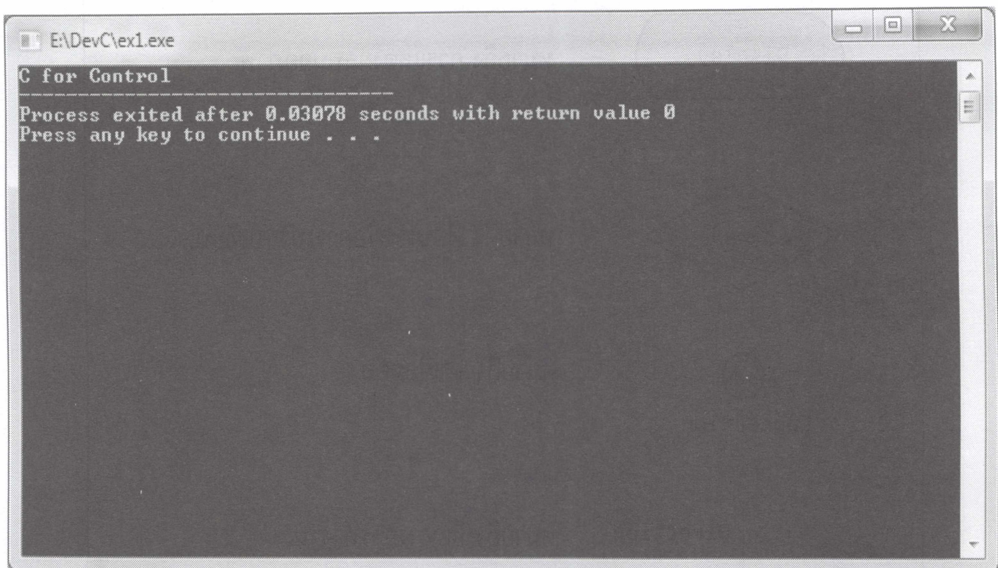
รูปที่ 1.9 ไฟล์ ex1.cpp

ลำดับที่ 7 ทำการ Compile และ Run โปรแกรม



รูปที่ 1.10 การ Compile และ Run โปรแกรม

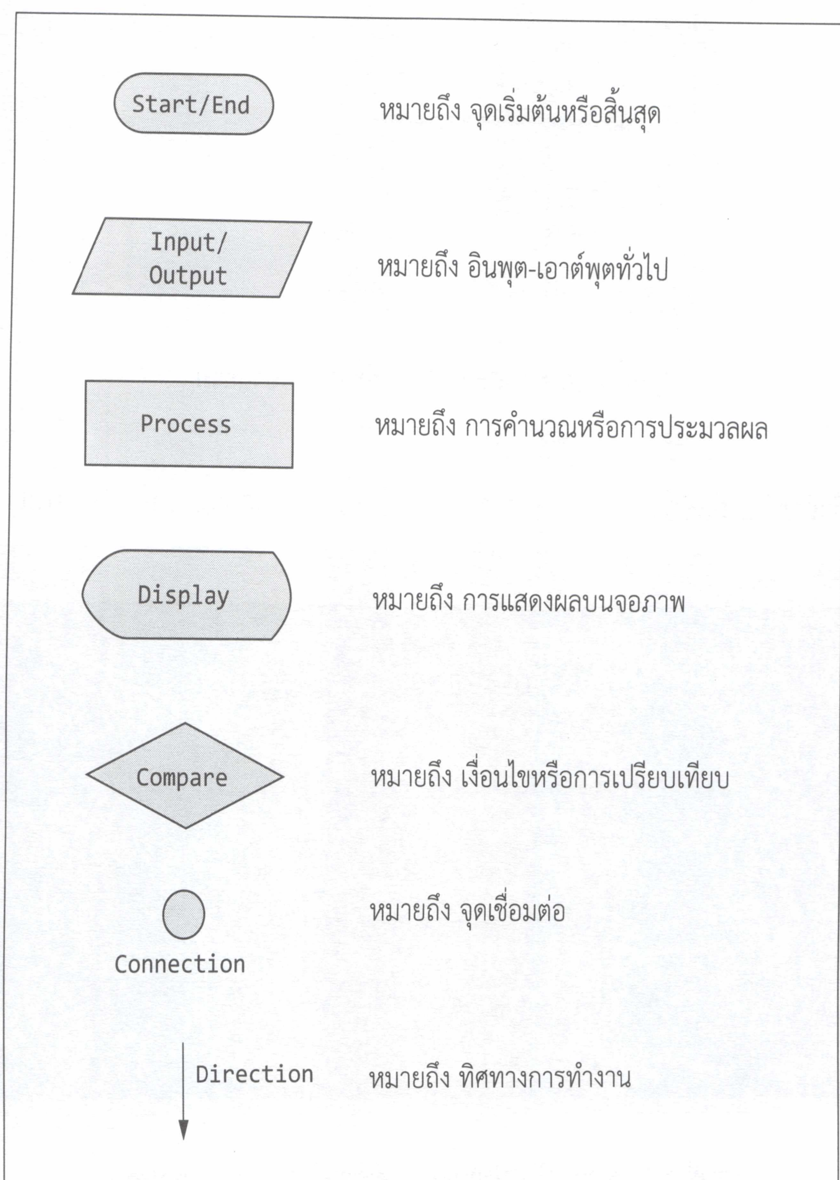
ลำดับที่ 8 ผลของการรันโปรแกรม หน้าจอจะแสดงข้อความ C for Control



รูปที่ 1.11 ผลการรันโปรแกรมแสดงข้อความ C for Control

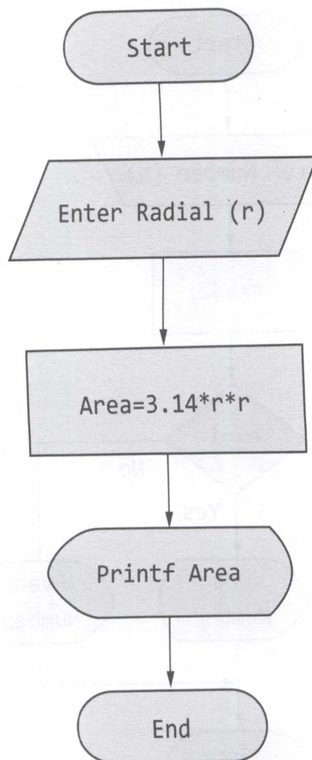
1.4 โฟลว์ชาร์ต

โฟลว์ชาร์ต (Flowchart) คือการเขียนแผนภาพที่แสดงลำดับขั้นตอนการทำงานของโปรแกรมหรือระบบงาน เพื่อให้ง่ายต่อการทำความเข้าใจและการแก้ไขปัญหา หลักการเขียนโฟลว์ชาร์ตควรเขียนจากบนลงล่าง มีจุดเริ่มต้นและสิ้นสุดเพียงจุดเดียว สัญลักษณ์พื้นฐานของการเขียนโฟลว์ชาร์ตแสดงดังรูปที่ 1.12



รูปที่ 1.12 สัญลักษณ์พื้นฐานของการเขียนโฟลว์ชาร์ต

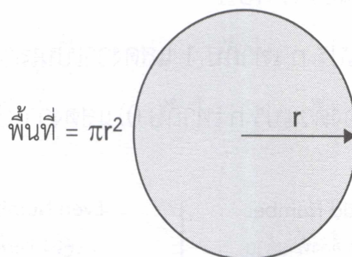
ตัวอย่างที่ 1.1 โฟลว์ชาร์ตการคำนวณหาพื้นที่วงกลม



รูปที่ 1.13 โฟลว์ชาร์ตการคำนวณหาพื้นที่วงกลม

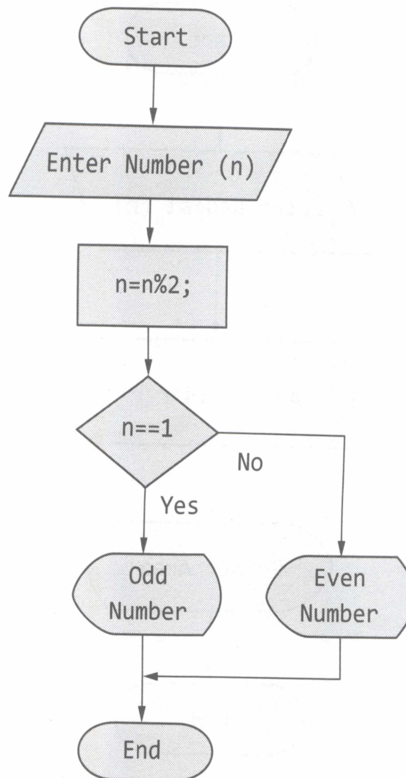
คำอธิบาย

1) เริ่มการทำงาน
2) รอรับค่ารัศมีมาเก็บไว้ที่ตัวแปร r
3) คำนวณหาพื้นที่ของวงกลมตามสูตร พื้นที่ = πr^2 หรือ 3.14*r*r



4) แสดงค่าพื้นที่ของวงกลมที่ได้จากการคำนวณ
5) จบการทำงาน

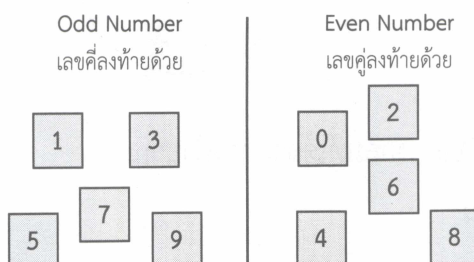
ตัวอย่างที่ 1.2 โฟลว์ชาร์ตการตรวจสอบเลขคู่-เลขคี่



รูปที่ 1.14 โฟลว์ชาร์ตการตรวจสอบเลขคู่-เลขคี่

คำอธิบาย

1) เริ่มการทำงาน
2) รอรับค่าจำนวนเต็มมาเก็บไว้ที่ตัวแปร n
3) คำนวณค่า n โดย n เท่ากับเศษที่ได้จากการหาร n ด้วย 2 ซึ่งผลลัพธ์คือ 0 หรือ 1
4) เปรียบเทียบค่าตัวแปร n กับ 1
    - ถ้าค่าของตัวแปร n เท่ากับ 1 แสดงว่าเป็นเลขคี่ ให้พิมพ์ Odd Number
    - ถ้าไม่ใช่ (ค่าของตัวแปร n เท่ากับ 0) แสดงว่าเป็นเลขคู่ ให้พิมพ์ Even Number



5) จบการทำงาน

1.5 ฟังก์ชัน printf();

ฟังก์ชัน printf(); เป็นฟังก์ชันที่ใช้พิมพ์หรือแสดงข้อความออกทางจอภาพ ถูกนิยามไว้ใน stdio.h

รูปแบบที่ 1

printf("ข้อความ");

ตัวอย่าง

printf("THAILAND");

ผลการรันโปรแกรม จะแสดงข้อความ THAILAND ออกทางจอภาพ

รูปแบบที่ 2

printf("ตัวกำหนดชนิดข้อมูล",ตัวแปร);

ตัวอย่าง

A=10;

printf("%d",A);

หมายถึง การนำค่าของตัวแปร A มาแสดงผลบนจอภาพในรูปของจำนวนเต็ม

ผลการรันโปรแกรม จะแสดงเลข 10 ออกทางจอภาพ

รูปแบบที่ 3

printf("ข้อความ ตัวกำหนดชนิดข้อมูล",ตัวแปร);

ตัวอย่าง

A=20;

printf("Data A = %d",A);

ผลการรันโปรแกรม จะแสดงข้อความ Data A = 20 ออกทางจอภาพ

1.6 ฟังก์ชัน scanf();

ฟังก์ชัน scanf(); เป็นฟังก์ชันที่ใช้รอรับค่าข้อมูลจากแป้นพิมพ์มาเก็บไว้ในตัวแปรเพื่อใช้สำหรับการคำนวณและการประมวลผล ซึ่งถูกนิยามไว้ใน stdio.h

รูปแบบ

scanf("ตัวกำหนดชนิดข้อมูล",ตัวแปร);

ตัวอย่าง

scanf("%d",&A);

หมายถึง รอรับค่าข้อมูลเป็นเลขจำนวนเต็มมาเก็บไว้ที่ตัวแปร A (&A คือที่อยู่ของตัวแปร A)

scanf("%f",&num);

หมายถึง รอรับค่าข้อมูลเป็นเลขทศนิยมมาเก็บไว้ที่ตัวแปร num

1.7 ตัวกำหนดชนิดข้อมูลของภาษาซี

ตารางที่ 1.1 ตัวกำหนดชนิดข้อมูลของภาษาซี

ตัวกำหนดชนิดข้อมูล	ความหมาย
%c	แสดงข้อมูลในรูปตัวอักษร
%d	แสดงข้อมูลในรูปจำนวนเต็ม
%e	แสดงข้อมูลในรูปเอกซ์โพเนนเชียล
%f	แสดงข้อมูลในรูปตัวเลขทศนิยม
%o	แสดงข้อมูลในรูปตัวเลขฐาน 8
%x	แสดงข้อมูลในรูปตัวเลขฐาน 16
%u	แสดงข้อมูลในรูปจำนวนเต็มบวก
%s	แสดงข้อมูลในรูปข้อความ
%p	แสดงข้อมูลในรูปที่อยู่ของตัวแปร

1.8 การประกาศตัวแปรในภาษาซี

การใช้งานตัวแปรในภาษาซีต้องมีการประกาศตัวแปรหรือการตั้งชื่อตัวแปรและกำหนดชนิดของตัวแปรจึงจะสามารถนำตัวแปรนั้นมาใช้งานได้ การตั้งชื่อตัวแปรมีข้อกำหนดดังนี้

1) ชื่อตัวแปรตัวแรกต้องขึ้นต้นด้วยตัวอักษรหรือเครื่องหมาย "_" เช่น num, max, _VR
2) ไม่มีการเว้นวรรคแต่สามารถใช้เครื่องหมาย "_" คั่นได้ เช่น sw_1, stop_motor
3) ถัดจากตัวแรกแล้วจะเป็นตัวเลขหรือเครื่องหมาย "_" ได้ เช่น a1, i_sensor, VR_1
4) อักษรตัวพิมพ์เล็กและพิมพ์ใหญ่ถือว่าแตกต่างกัน เช่น a1, A1 เป็นคนละตัวแปร
5) ห้ามตั้งชื่อซ้ำกับคำสงวนในภาษาซี อย่างเช่น if, do, while, printf, else ฯลฯ

ตัวอย่างการตั้งชื่อตัวแปรที่ถูกต้อง

```
int  a1, max, num;
char c, VR_1, sw1;
```

ตัวอย่างการตั้งชื่อตัวแปรที่ผิด

```
int  1a, ma x, 2num;
char else, for, data 1;
```

1.9 ชนิดของตัวแปรในภาษาซี

ตารางที่ 1.2 ชนิดของตัวแปรในภาษาซี

ชนิดของตัวแปร	ขนาด	ช่วงของข้อมูล	การกำหนดชนิดตัวแปร
1) Character	8 บิต	-128 ถึง +127	char
2) Unsigned Character	8 บิต	0 ถึง 255	unsigned char
3) Integer	16 บิต	-32,768 ถึง +32,767	int
4) Unsigned Integer	16 บิต	0 ถึง 65,535	unsigned int
5) Long Integer	32 บิต	-2,147,483,648 ถึง 2,147,483,647	long int
6) Floating Point	32 บิต	3.4×10^{-38} ถึง $3.4 \times 10^{+38}$	float
7) Double	64 บิต	1.7×10^{-308} ถึง $1.7 \times 10^{+308}$	double

1) Character ตัวแปรชนิดตัวอักษรหรือจำนวนเต็มขนาด 8 บิต ใช้เก็บข้อมูลที่เป็นตัวอักษรหรือจำนวนเต็มที่มีค่าอยู่ในช่วง -128 ถึง +127

2) Unsigned Character ตัวแปรชนิดตัวอักษรหรือจำนวนเต็มขนาด 8 บิต แบบไม่คิดเครื่องหมาย ใช้เก็บข้อมูลที่เป็นตัวอักษรหรือจำนวนเต็มบวกที่มีค่าอยู่ในช่วง 0 ถึง 255

3) Integer ตัวแปรชนิดจำนวนเต็มขนาด 16 บิต ใช้เก็บข้อมูลที่เป็นจำนวนเต็มที่มีค่าอยู่ในช่วง -32,768 ถึง +32,767

4) Unsigned Integer ตัวแปรชนิดจำนวนเต็มขนาด 16 บิต แบบไม่คิดเครื่องหมายหรือจำนวนเต็มบวก ใช้เก็บจำนวนเต็มบวกที่มีค่าอยู่ในช่วง 0 ถึง 65,535

5) Long Integer ตัวแปรชนิดจำนวนเต็มขนาด 32 บิต ใช้เก็บข้อมูลจำนวนเต็มที่มีค่าอยู่ในช่วง -2,147,483,648 ถึง +2,147,483,647

6) Floating Point ตัวแปรชนิดเลขทศนิยมขนาด 32 บิต ใช้เก็บข้อมูลเลขทศนิยมที่มีค่าอยู่ในช่วง 3.4×10^{-38} ถึง $3.4 \times 10^{+38}$

7) Double ตัวแปรชนิดเลขทศนิยมขนาด 64 บิต ใช้เก็บข้อมูลเลขทศนิยมที่มีค่าอยู่ในช่วง 1.7×10^{-308} ถึง $1.7 \times 10^{+308}$

ตัวอย่างการประกาศตัวแปร

char key; หมายถึง การประกาศตัวแปร key เป็นตัวแปรชนิดตัวอักษรหรือจำนวนเต็มขนาด 8 บิต

int A, B; หมายถึง การประกาศตัวแปร A และ B เป็นตัวแปรชนิดจำนวนเต็มขนาด 16 บิต

float f1, f2; หมายถึง การประกาศตัวแปร f1 และ f2 เป็นตัวแปรชนิดเลขทศนิยมเพื่อใช้คำนวณตัวเลขที่มีจุดทศนิยม

long int l1, num; หมายถึง การประกาศตัวแปร l1 และ num เป็นตัวแปรชนิดจำนวนเต็มขนาด 32 บิต ใช้ในการคำนวณตัวเลขที่มีค่ามาก ๆ

การกำหนดชนิดของตัวแปรควรกำหนดให้สอดคล้องกับการใช้งาน อย่างเช่น ถ้ามีการคำนวณตัวเลขที่เป็นจุดทศนิยม ต้องกำหนดชนิดของตัวแปรเป็นแบบ float หรือการกำหนดตัวแปรเพื่อใช้ในการวนรอบที่ไม่เกิน 127 รอบ ควรกำหนดตัวแปรเป็นแบบ char เป็นต้น ทั้งนี้เพื่อลดขนาดของโปรแกรมให้เล็กลงและทำให้การประมวลผลเร็วขึ้น

1.10 ตัวดำเนินการทางคณิตศาสตร์

ในการเขียนโปรแกรมที่มีการคำนวณทางคณิตศาสตร์ จำเป็นต้องทราบความหมายของตัวดำเนินการทางคณิตศาสตร์ตามลักษณะโครงสร้างของภาษาซี เพื่อให้สามารถเขียนโปรแกรมคำนวณได้อย่างถูกต้อง

ตารางที่ 1.3 ตัวดำเนินการทางคณิตศาสตร์

ตัวดำเนินการทางคณิตศาสตร์	ความหมาย
+	การบวก
-	การลบ
*	การคูณ
/	การหาร
%	การหารที่คิดเฉพาะเศษ
++	การเพิ่มค่าขึ้น 1
--	การลดค่าลง 1

ตัวอย่างการใช้ตัวดำเนินการทางคณิตศาสตร์

a = b+5; หมายถึง a เท่ากับ b บวก 5

c = a-2; หมายถึง c เท่ากับ a ลบ 2

x = a*b; หมายถึง x เท่ากับ a คูณ b

y = a/b; หมายถึง y เท่ากับ a หาร b

z = 7%2; หมายถึง z เท่ากับ เศษที่ได้จากการหาร 7 ด้วย 2 ซึ่งจะได้ผลลัพธ์คือ 1

a++; หรือ a = a+1; หมายถึง เพิ่มค่า a ครั้งละ 1

a--; หรือ a = a-1; หมายถึง ลดค่า a ครั้งละ 1

ตัวอย่างที่ 1.3 โปรแกรมแสดงค่าตามตัวกำหนดชนิดข้อมูล

```
#include <stdio.h>
int main()
{
  char x=65;
  printf(“char x = %c \n”,x);
  printf(“int  x = %d \n”,x);
  printf(“Hex  x = %x \n”,x);
  printf(“float x = %f \n”,x);
}
```

ผลการรันโปรแกรม

โปรแกรมจะแสดงค่าตามตัวกำหนดชนิดข้อมูลดังนี้

char x = A // แสดงเป็นตัวอักษร

// (65 เลขฐานสิบจะมีค่าเท่ากับตัวอักษร A ตามรหัสแอสกี)

int x = 65 // แสดงเป็นตัวเลข

Hex x = 41 // แสดงเป็นเลขฐาน 16

float x = 0.000000 // แสดงเป็นเลขทศนิยม จะแสดงได้ก็ต่อเมื่อประกาศตัวแปร float

หมายเหตุ : *รหัสแอสกี (ASCII : American Standard Code for Information Interchange) เป็นรหัสมาตรฐานของข้อมูลเพื่อใช้สื่อสารในระบบคอมพิวเตอร์ ซึ่งประกอบไปด้วยตัวอักษร ตัวเลข เครื่องหมาย และสัญลักษณ์ต่าง ๆ โดยแต่ละรหัสจะแทนด้วยหนึ่งอักขระ ตัวอย่างเช่น 65 (เลขฐานสิบ) ใช้แทนอักษร A สามารถดูรายละเอียดเพิ่มเติมได้จากตารางรหัสแอสกี*

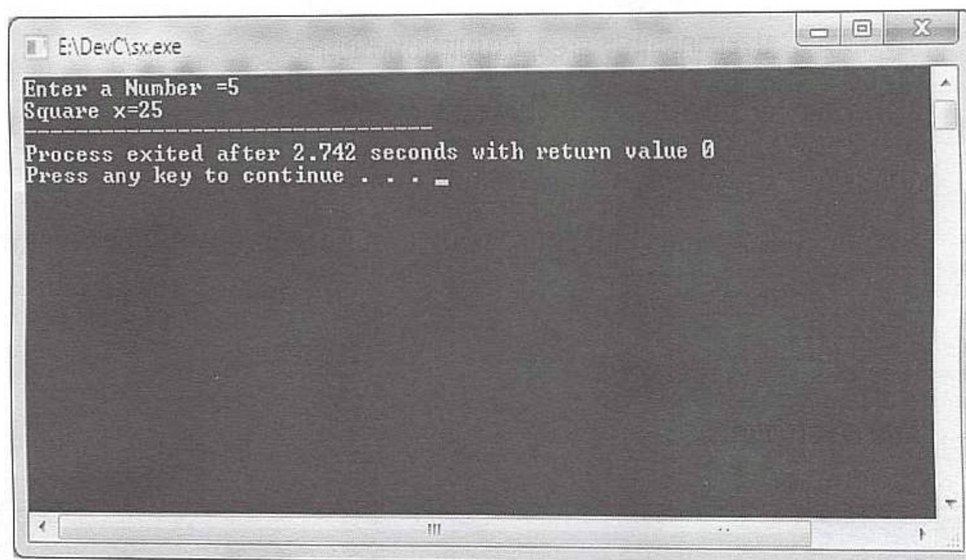
ตัวอย่างที่ 1.4 โปรแกรมคำนวณหาค่าของเลขยกกำลัง

```
#include <stdio.h>
int main()
{
 int x,sx;
 printf("Enter a Number =");
 scanf("%d",&x);
 sx=x*x;
 printf("Square x=%d",sx);
}
```

```
// ไฟล์ส่วนหัวโปรแกรม
// ฟังก์ชันหลักของโปรแกรม
// ปีกกาเปิด เริ่มต้นเขียนโปรแกรม
// ประกาศตัวแปร x, sx เป็นชนิดจำนวนเต็ม
// พิมพ์หรือแสดงข้อความ
// รอรับค่าตัวเลขมาเก็บไว้ในตัวแปร x
// คำนวณค่ายกกำลัง โดย sx = x*x
// แสดงข้อความและผลของการยกกำลัง
// ปีกกาปิด จบโปรแกรม
```

ผลการรันโปรแกรม

โปรแกรมจะรอรับค่าตัวเลขจากแป้นพิมพ์ จากนั้นนำตัวเลขมายกกำลังหรือคูณกันแล้วแสดงผลการคำนวณบนจอภาพ



รูปที่ 1.15 ผลการรันโปรแกรมคำนวณเลขยกกำลัง

ตัวอย่างที่ 1.5 โปรแกรมคำนวณหาพื้นที่วงกลม

```
1  #include <stdio.h>
2  int main()
3  {
4     float area,r;
5     printf("Enter a Radial =");
6     scanf("%f",&r);
7     area=3.14*r*r;
8     printf("Area a Circle =%.2f",area);
9  }
```

คำอธิบาย

บรรทัดที่ 1 ประกาศไฟล์ส่วนหัวโปรแกรม เพื่อให้สามารถเรียกใช้ฟังก์ชันต่าง ๆ ของ stdio.h ได้ เช่น printf(); scanf();

บรรทัดที่ 2 ฟังก์ชันหลักของโปรแกรม

บรรทัดที่ 3 ปีกกาเปิด เริ่มต้นเขียนโปรแกรม

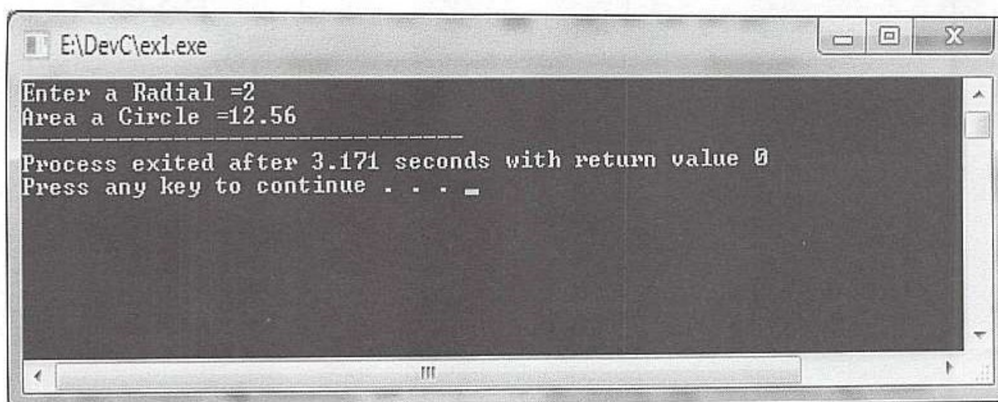
บรรทัดที่ 4 ประกาศตัวแปร area, r เป็นแบบทศนิยม

บรรทัดที่ 6 รอรับค่ารัศมีมาเก็บไว้ในตัวแปร r

บรรทัดที่ 7 คำนวณหาพื้นที่วงกลมตามสูตร พื้นที่ = 3.14 × r × r

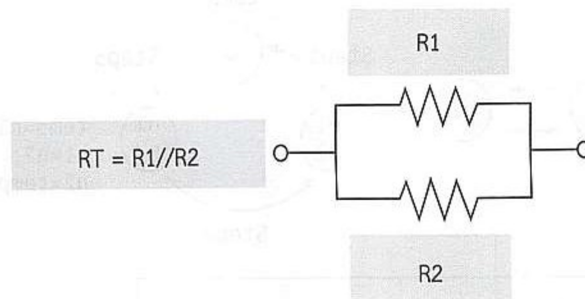
บรรทัดที่ 8 แสดงข้อความและผลการคำนวณพื้นที่วงกลมเป็นทศนิยมสองตำแหน่ง (%.2f)

ผลการรันโปรแกรม



รูปที่ 1.16 ผลการรันโปรแกรมคำนวณหาพื้นที่วงกลม

ตัวอย่างที่ 1.6 โปรแกรมคำนวณหาค่าความต้านทาน R1 ขนานกับ R2




```
1  #include <stdio.h>
2  int main()
3  { float rt,r1,r2;
4    printf("Enter R1 R2 =");
5    scanf("%f %f",&r1,&r2);
6    rt=r1*r2/(r1+r2);
7    printf("RT=R1//R2=%.2f",rt);
8  }
```

คำอธิบาย

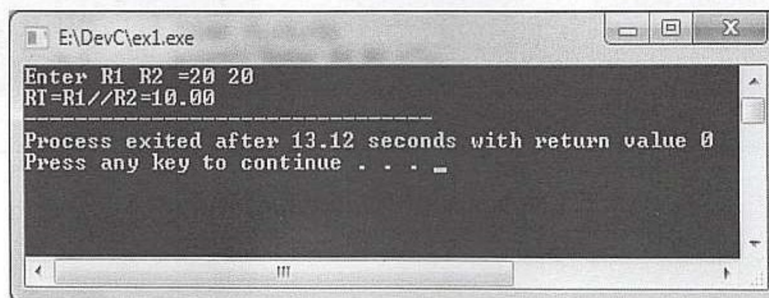
บรรทัดที่ 3 ประกาศตัวแปร rt, r1, r2 เป็นแบบทศนิยม

บรรทัดที่ 5 รอรับค่าความต้านทานมาเก็บไว้ในตัวแปร r1 และ r2

บรรทัดที่ 6 คำนวณหาค่าความต้านทานรวมตามสูตร

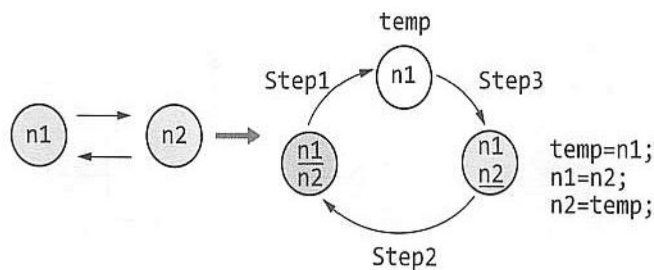
บรรทัดที่ 7 แสดงข้อความและผลการคำนวณค่าความต้านทานรวม

ผลการรันโปรแกรม



รูปที่ 1.17 ผลการรันโปรแกรมคำนวณหาค่าความต้านทาน R1 ขนานกับ R2

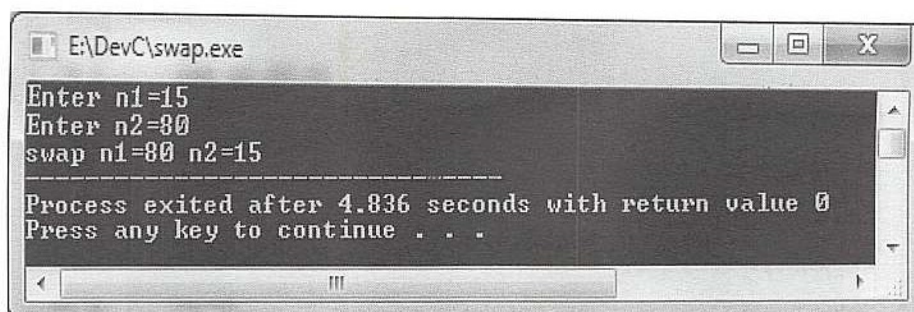
ตัวอย่างที่ 1.7 โปรแกรมสลับค่าตัวเลข




```
#include <stdio.h>
int main()
{
  int n1,n2,temp;
  printf(“Enter n1=”);
  scanf(“%d”,&n1);
  printf(“Enter n2=”);
  scanf(“%d”,&n2);
  temp=n1;
  n1=n2;
  n2=temp;
 printf(“swap n1=%d n2=%d”,n1,n2);
}
```

ผลการรันโปรแกรม

โปรแกรมจะสลับค่าระหว่างตัวแปร n1 และ n2 โดยมีตัวแปร temp ไว้พักค่าของตัวแปร n1



รูปที่ 1.18 ผลการรันโปรแกรมสลับค่าตัวเลข

1.11 สรุป

ภาษาซี เป็นภาษาที่ได้รับความนิยมมาตั้งแต่อดีตจนถึงปัจจุบัน จัดเป็นภาษาที่มีความยืดหยุ่นในการเขียนโปรแกรม มีลักษณะของภาษาที่เป็นโครงสร้าง และการเขียนโปรแกรมเป็นแบบลำดับ ภาษาซีสามารถเขียนโปรแกรมเพื่อควบคุมไมโครคอนโทรลเลอร์ในตระกูลต่าง ๆ ได้ เช่น MCS-51 PIC และ Arduino ส่วนประกอบของภาษาซีแบ่งออกได้เป็น 2 ส่วนใหญ่ ๆ คือ ไฟล์ส่วนหัวโปรแกรมและตัวโปรแกรม

1) ไฟล์ส่วนหัวโปรแกรม เป็นไฟล์ที่มีส่วนขยายเป็น *.h อย่างเช่น stdio.h เป็นไฟล์ที่ใช้เก็บไลบรารีมาตรฐานของภาษาซีเกี่ยวกับการรับข้อมูลและการแสดงผล ซึ่งต้องประกาศไฟล์ stdio.h เพื่อใช้ร่วมในการคอมไพล์โปรแกรม

2) ตัวโปรแกรม เริ่มต้นด้วยฟังก์ชัน int main() ซึ่งเป็นฟังก์ชันหลัก โดยมีเครื่องหมายปีกกาเปิดและปิดแสดงการเริ่มต้นและจบโปรแกรม ภายในฟังก์ชัน int main() จะประกอบด้วยชุดคำสั่งและฟังก์ชันต่าง ๆ

การเขียนโปรแกรมภาษาซีเพื่อควบคุมการทำงานของไมโครคอนโทรลเลอร์ให้สามารถทำงานได้จะต้องเรียนรู้เกี่ยวกับโปรแกรมภาษาซี พื้นฐานของไมโครคอนโทรลเลอร์ Arduino และการเชื่อมต่อกับอุปกรณ์อินพุต เอาต์พุต จึงจะสามารถประยุกต์ใช้ในงานระบบควบคุมต่าง ๆ ได้

คำถามท้ายบทที่ 1

1. จงบอกประวัติของภาษาซี
2. จงบอกส่วนประกอบของภาษาซี
3. จงอธิบายความแตกต่างของตัวแปร unsigned char และ char
4. จงอธิบายความแตกต่างของตัวแปร int และ float
5. a=5%2; และ a=5/2; ต่างกันอย่างไร
6. จงอธิบายหลักการสลับข้อมูลของตัวอย่างที่ 1.7
7. จงเขียนโฟลว์ชาร์ตการตัดเกรด A B C D และ E
8. จงเขียนโปรแกรมหาค่าพื้นที่สี่เหลี่ยมคางหมู



การเขียนโปรแกรมเปรียบเทียบ

การตัดสินใจทำงานของโปรแกรมหรือเครื่องจักรกลที่ควบคุมด้วยคอมพิวเตอร์ เพื่อให้ทำงานอย่างใดอย่างหนึ่งตามเงื่อนไขหรืออัลกอริทึมที่กำหนดไว้ว่าจะทำส่วนใดก่อนหรือจะทำหรือไม่ เช่น ถ้ามีการกดสวิตช์ให้มอเตอร์ทำงาน หรือถ้าความเข้มของแสงน้อยกว่าค่าที่กำหนดไว้แสดงว่ามืดให้เปิดไฟ เงื่อนไขต่าง ๆ เหล่านี้จำเป็นต้องใช้ฟังก์ชันเปรียบเทียบข้อมูล ซึ่งในภาษาซีมีฟังก์ชัน if() เป็นฟังก์ชันที่ใช้ในการเปรียบเทียบข้อมูล มีทั้งแบบทางเลือกเดียวและหลายทางเลือก ถ้าหากเงื่อนไขในการเปรียบเทียบเป็นจริงก็จะทำในชุดฟังก์ชันที่ 1 แต่ถ้าเงื่อนไขเป็นเท็จก็จะทำชุดฟังก์ชันที่ 2 นอกจากฟังก์ชัน if() แล้วยังมีฟังก์ชัน switch() เป็นฟังก์ชันที่ใช้ในการเปรียบเทียบซึ่งเหมาะกับการเปรียบเทียบข้อมูลที่มีหลายทางเลือก

2.1 ฟังก์ชัน if() ทางเลือกเดียว

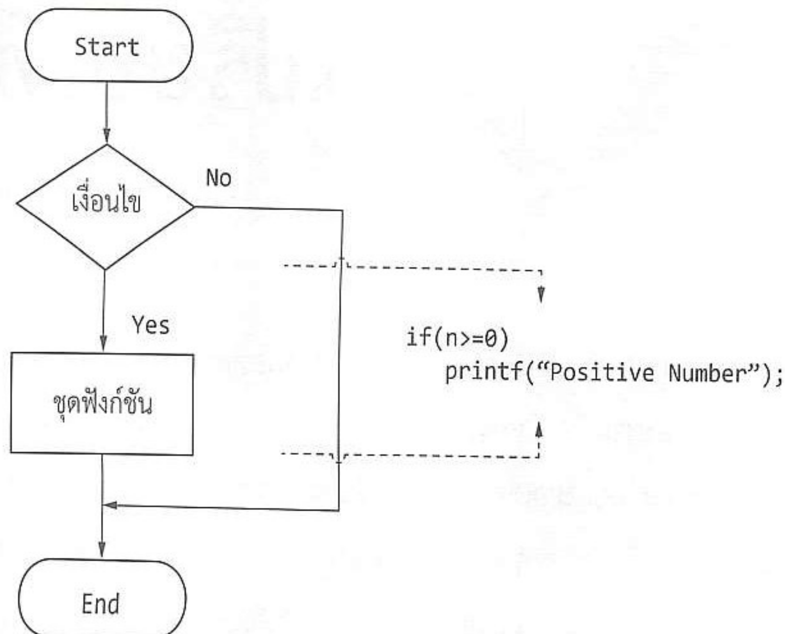
การทำงานของฟังก์ชัน if() ทางเลือกเดียวคือ ถ้าเงื่อนไขในการเปรียบเทียบเป็นจริงจะทำชุดฟังก์ชันในปีกกาเปิดและปิด แต่ถ้าเงื่อนไขไม่เป็นจริงหรือเป็นเท็จจะจบการทำงาน

รูปแบบ

```
if(ตัวแปร ตัวดำเนินการเปรียบเทียบ ค่าคงที่หรือตัวแปร)
 {
   ชุดฟังก์ชัน
 }
```

ตัวอย่าง การทำงานของฟังก์ชัน if() ทางเลือกเดียว

```
if(n>=0)
    printf("Positive Number");
```



รูปที่ 2.1 การทำงานของฟังก์ชัน if() ทางเลือกเดียว

จากตัวอย่างด้านบน หมายถึง ถ้าตัวแปร n มีค่ามากกว่าหรือเท่ากับ 0 ให้แสดงข้อความ Positive Number ในฟังก์ชัน ถ้ามีมากกว่าหนึ่งฟังก์ชันหรือหนึ่งคำสั่งต้องใส่เครื่องหมายปีกกาเปิดและปิด แต่ถ้ามีเพียงฟังก์ชันเดียวหรือคำสั่งเดียวไม่ต้องใส่เครื่องหมายปีกกาเปิดและปิดก็ได้

ข้อควรระวัง

เครื่องหมายปีกกาเปิดและปิดในฟังก์ชัน if() ถ้าไม่ใส่ โปรแกรมจะทำฟังก์ชันเดียวเท่านั้น และฟังก์ชัน if() จะไม่มีเครื่องหมายเซมิโคลอนปิดท้าย ถ้าหากใส่เครื่องหมายเซมิโคลอนปิดท้ายฟังก์ชัน if() ในการรันและการคอมไพล์จะไม่แจ้งข้อผิดพลาด แต่ผลการรันโปรแกรมจะไม่เป็นไปตามเงื่อนไขที่ต้องการ

2.2 ตัวดำเนินการเปรียบเทียบ

ตัวดำเนินการเปรียบเทียบใช้สำหรับเปรียบเทียบข้อมูล เพื่อเป็นเงื่อนไขในการทำงานของโปรแกรม โดยตัวดำเนินการเปรียบเทียบแสดงดังตารางที่ 2.1

ตารางที่ 2.1 ตัวดำเนินการเปรียบเทียบ

สัญลักษณ์	ความหมาย
>	มากกว่า
>=	มากกว่าหรือเท่ากับ
<	น้อยกว่า
<=	น้อยกว่าหรือเท่ากับ
==	เท่ากับหรือเท่ากัน
!=	ไม่เท่ากับหรือไม่เท่ากัน

ตัวอย่าง การใช้งานตัวดำเนินการเปรียบเทียบ

```
if(a>b)
    max=a;
```

หมายถึง ถ้าตัวแปร a มีค่ามากกว่า b ให้ตัวแปร max = a

2.3 ตัวดำเนินการลอจิก

ตารางที่ 2.2 ตัวดำเนินการลอจิก

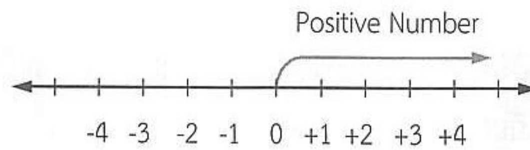
สัญลักษณ์	ความหมาย	
&&	AND	และ
||	OR	หรือ
!	NOT	ไม่

ตัวอย่าง การใช้งานตัวดำเนินการลอจิก

```
if(a>b && a>c)
    max=a;
```

หมายถึง ถ้าตัวแปร a มีค่ามากกว่า b และ c ให้ตัวแปร max = a

ตัวอย่างที่ 2.1 โปรแกรมตรวจสอบเลขจำนวนเต็มบวก



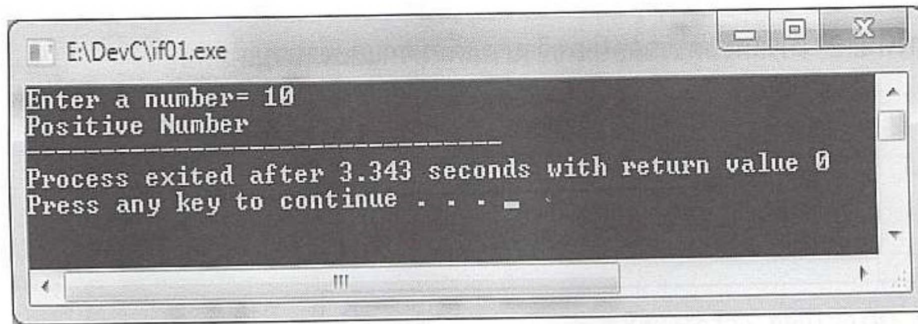

```
1  #include <stdio.h>
2  int main()
3   {
4     int number;
5     printf("Enter a number= ");
6     scanf("%d",&number);
7     if(number>=0)
8        printf("Positive Number");
9   }
```

คำอธิบาย

บรรทัดที่ 1	ประกาศไฟล์ส่วนหัวโปรแกรมเพื่อให้สามารถเรียกใช้ฟังก์ชันต่าง ๆ ของ stdio.h ได้ เช่น printf(); scanf();
บรรทัดที่ 2	ฟังก์ชันหลักของโปรแกรม
บรรทัดที่ 3	ปีกกาเปิด เริ่มต้นเขียนโปรแกรม
บรรทัดที่ 4	ประกาศตัวแปร number เป็นชนิดจำนวนเต็ม
บรรทัดที่ 5	แสดงข้อความ Enter a number=
บรรทัดที่ 6	รอรับค่าจำนวนเต็มมาเก็บไว้ที่ตัวแปร number
บรรทัดที่ 7 และ 8	ถ้าตัวแปร number มากกว่าหรือเท่ากับ 0 ให้แสดงข้อความ Positive Number
บรรทัดที่ 9	ปีกกาปิด จบโปรแกรม

ผลการรันโปรแกรม

โปรแกรมจะรอรับค่าตัวเลขมาเก็บไว้ที่ตัวแปร number จากนั้นเปรียบเทียบค่าตัวเลขที่รับเข้ามาโดยฟังก์ชัน if() ถ้ามีค่ามากกว่าหรือเท่ากับศูนย์จะแสดงข้อความ Positive Number



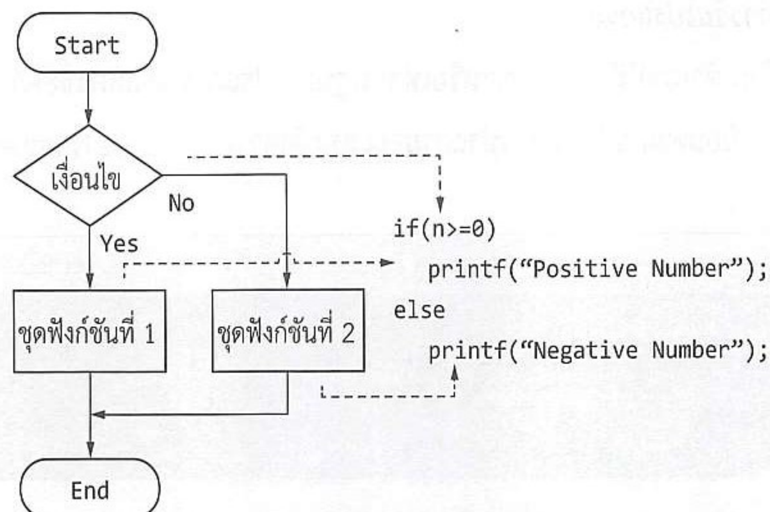
รูปที่ 2.2 ผลการรันโปรแกรมตรวจสอบเลขจำนวนเต็มบวก

2.4 ฟังก์ชัน if() สองทางเลือก

ฟังก์ชัน if() สองทางเลือกต้องใช้ร่วมกับ else การทำงานคือ ถ้าเงื่อนไขในการเปรียบเทียบเป็นจริงจะทำชุดฟังก์ชันที่ 1 แต่ถ้าเงื่อนไขเป็นเท็จจะทำชุดฟังก์ชันที่ 2

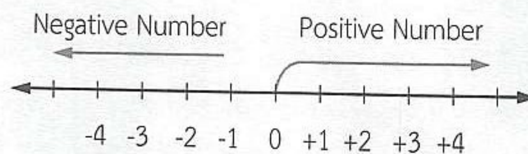
รูปแบบ

```
if(ตัวแปร ตัวดำเนินการเปรียบเทียบ ค่าคงที่หรือตัวแปร)
  {
    ชุดฟังก์ชันที่ 1
  }
else
  {
    ชุดฟังก์ชันที่ 2
  }
```



รูปที่ 2.3 โฟลว์ชาร์ตการทำงานของฟังก์ชัน if() สองทางเลือก

ตัวอย่างที่ 2.2 โปรแกรมตรวจสอบเลขจำนวนเต็มบวกและจำนวนเต็มลบ




```
1   #include <stdio.h>
2   int main()
3    {
4      int number;
5        printf("Enter a number: ");
6        scanf("%d",&number);
7        if(number>=0)
8          printf("Positive Number");
9        else
10         printf("Negative Number");
11   }
```

คำอธิบาย

บรรทัดที่ 5 แสดงข้อความ Enter a number:

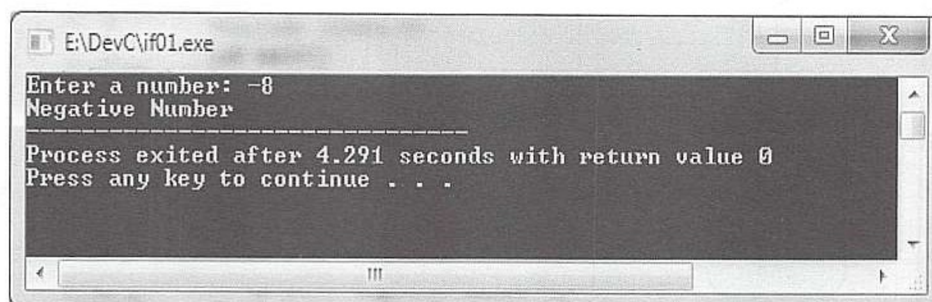
บรรทัดที่ 6 รอรับค่าจำนวนเต็มมาเก็บไว้ที่ตัวแปร number

บรรทัดที่ 7 และ 8 ถ้าตัวแปร number ≥ 0 หรือถ้าเงื่อนไขเป็นจริงให้แสดงข้อความ Positive Number

บรรทัดที่ 9 และ 10 ถ้าเป็นเท็จให้แสดงข้อความ Negative Number

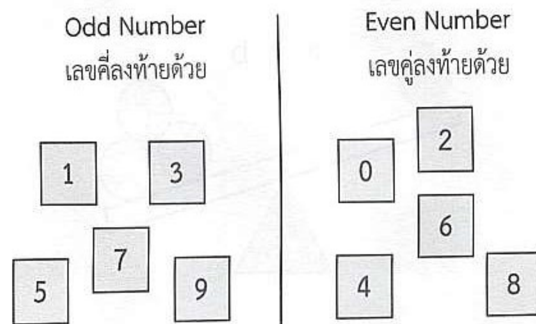
ผลการรันโปรแกรม

ถ้าป้อนตัวเลขที่มีค่ามากกว่าหรือเท่ากับศูนย์ โปรแกรมจะแสดงข้อความ Positive Number แต่ถ้าป้อนตัวเลขที่เป็นลบ โปรแกรมจะแสดงข้อความ Negative Number



รูปที่ 2.4 ผลการรันโปรแกรมตรวจสอบเลขจำนวนเต็มบวกและจำนวนเต็มลบ

ตัวอย่างที่ 2.3 โปรแกรมตรวจสอบเลขคู่-เลขคี่




```
1   #include <stdio.h>
2   int main()
3    {
4     int number;
5     printf("Enter a number: ");
6     scanf("%d",&number);
7     if(number%2==1)
8       printf("Odd Number");
9     else
10      printf("Even Number");
11   }
```

คำอธิบาย

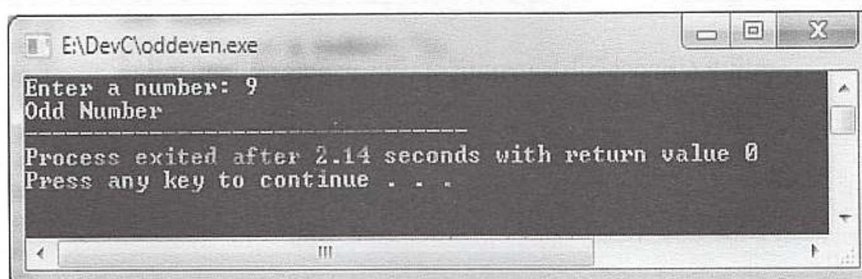
บรรทัดที่ 6 รอรับค่าจำนวนเต็มมาเก็บไว้ในตัวแปร number

บรรทัดที่ 7 และ 8 ถ้าเศษจากการหารด้วย 2 เท่ากับ 1 ให้แสดงข้อความ Odd Number

บรรทัดที่ 9 และ 10 ถ้าไม่ใช่ ให้แสดงข้อความ Even Number

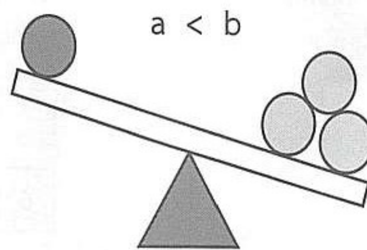
ผลการรันโปรแกรม

เมื่อป้อนเลขคี่ โปรแกรมจะแสดงข้อความ Odd Number และเมื่อป้อนเลขคู่ โปรแกรมจะแสดงข้อความ Even Number



รูปที่ 2.5 ผลการรันโปรแกรมตรวจสอบเลขคู่-เลขคี่

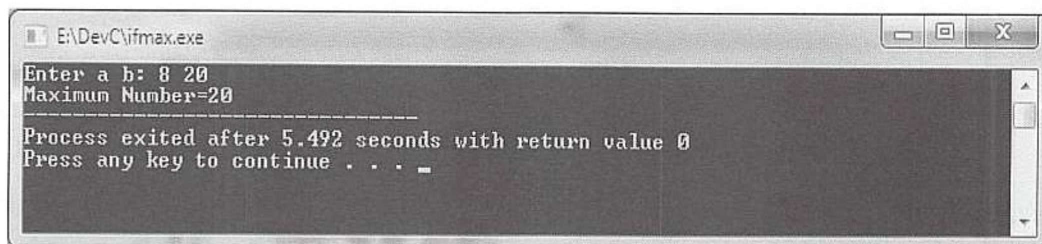
ตัวอย่างที่ 2.4 โปรแกรมเปรียบเทียบหาตัวเลขที่มีค่ามากที่สุด




```
#include <stdio.h>
int main()
 {
  int max,a,b;
  printf(“Enter a b: ”);
  scanf(“%d %d”,&a,&b);
    if(a>b)
     max=a;
    else
     max=b;
  printf(“Maximum Number=%d”,max);
 }
```

ผลการรันโปรแกรม

โปรแกรมจะรับเลขสองจำนวนมาเก็บไว้ในตัวแปร a และ b จากนั้นนำมาเปรียบเทียบเพื่อหาค่ามากที่สุดมาเก็บไว้ในตัวแปร max แล้วแสดงผลค่าที่มากที่สุด



รูปที่ 2.6 ผลการรันโปรแกรมเปรียบเทียบหาตัวเลขที่มีค่ามากที่สุด

ตัวอย่างที่ 2.5 โปรแกรมตรวจสอบตัวอักษรภาษาอังกฤษ (ทั้งตัวพิมพ์เล็กและพิมพ์ใหญ่)

a b c...z or A B C...Z

```
1  #include <stdio.h>
2  int main()
3   {
4     char c;
5     printf(“Enter a Character: ”);
6     scanf(“%c”,&c);
7     if((c>=‘a’&&c<=‘z’)||(c>=‘A’&&c<=‘Z’))
8      printf(“%c is an Alphabet.”,c);
9     else
10     printf(“%c is not an Alphabet.”,c);
11  }
```

คำอธิบาย

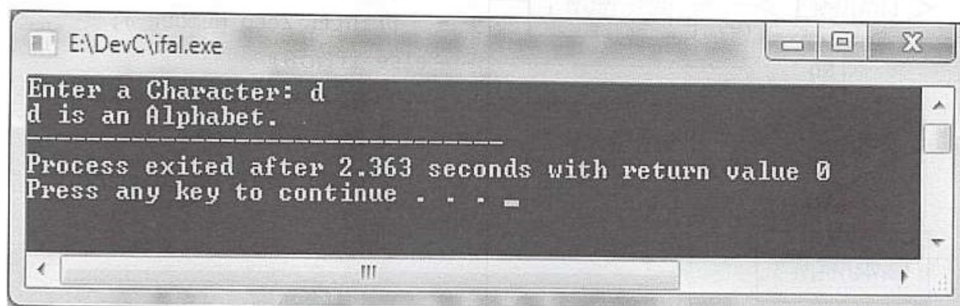
บรรทัดที่ 6 รอรับค่าตัวอักษรจากแป้นพิมพ์มาเก็บไว้ในตัวแปร c

บรรทัดที่ 7 และ 8 ถ้าตัวแปร c เป็นตัวอักษร a-z หรือ A-Z ให้แสดงข้อความว่า (ตัวแปร c) is an Alphabet.

บรรทัดที่ 9 และ 10 ถ้าไม่ใช่ ให้แสดงข้อความว่า (ตัวแปร c) is not an Alphabet.

ผลการรันโปรแกรม

เมื่อป้อนตัวอักษรภาษาอังกฤษ โปรแกรมจะแสดงข้อความว่า ตัวแปร c เป็นตัวอักษร แต่ถ้าป้อนตัวเลขหรือสัญลักษณ์อื่น โปรแกรมจะแสดงข้อความว่า ตัวแปร c ไม่ใช่ตัวอักษร



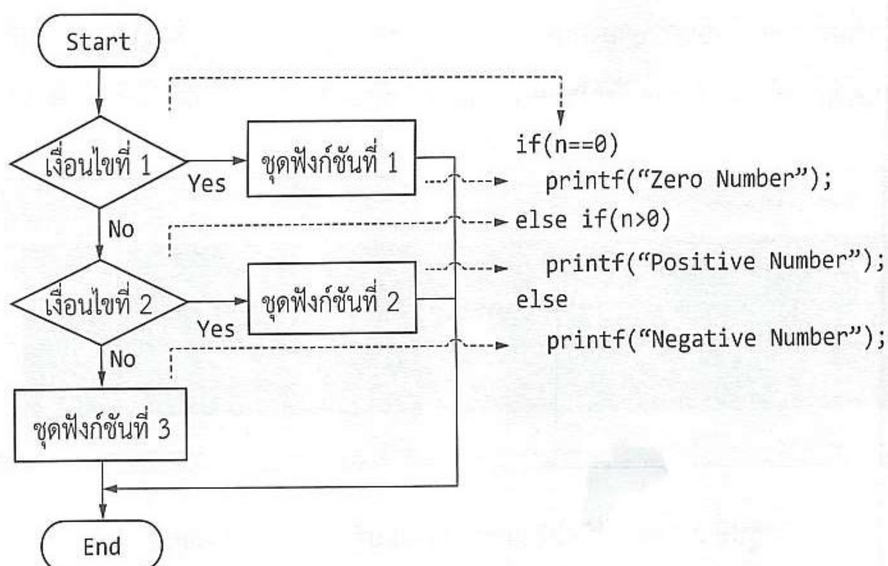
รูปที่ 2.7 ผลการรันโปรแกรมตรวจสอบตัวอักษรภาษาอังกฤษ

2.5 ฟังก์ชัน if() หลายทางเลือก

ฟังก์ชัน if() หลายทางเลือก จะทำการเปรียบเทียบหลาย ๆ เงื่อนไข ถ้าเงื่อนไขที่ 1 เป็นจริงจะทำชุดฟังก์ชันที่ 1 แต่ถ้าเงื่อนไขเป็นเท็จจะทำการเปรียบเทียบกับเงื่อนไขที่ 2 ถ้าเงื่อนไขที่ 2 เป็นจริงจะทำชุดฟังก์ชันที่ 2 แต่ถ้าเป็นเท็จจะทำชุดฟังก์ชันที่ 3

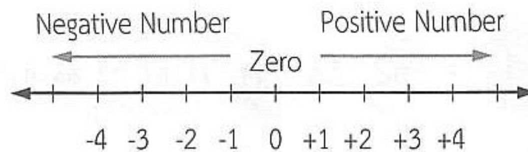
รูปแบบ

```
if(ตัวแปร ตัวดำเนินการเปรียบเทียบ ค่าคงที่หรือตัวแปร)
  {
    ชุดฟังก์ชันที่ 1
  }
else if(ตัวแปร ตัวดำเนินการเปรียบเทียบ ค่าคงที่หรือตัวแปร)
  {
    ชุดฟังก์ชันที่ 2
  }
else
  {
    ชุดฟังก์ชันที่ 3
  }
```



รูปที่ 2.8 โฟลว์ชาร์ตการทำงานของฟังก์ชัน if() หลายทางเลือก

ตัวอย่างที่ 2.6 โปรแกรมตรวจสอบเลขจำนวนเต็มบวก จำนวนเต็มลบ หรือศูนย์




```
1   #include <stdio.h>
2   int main()
3    {
4      int number;
5      printf("Enter a number: ");
6      scanf("%d",&number);
7        if(number==0)
8          printf("Zero Number");
9        else if(number>0)
10         printf("Positive Number");
11       else
12         printf("Negative Number");
13   }
```

คำอธิบาย

บรรทัดที่ 5 แสดงข้อความ Enter a number:

บรรทัดที่ 6 รอรับค่าจำนวนเต็มมาเก็บไว้ที่ตัวแปร number

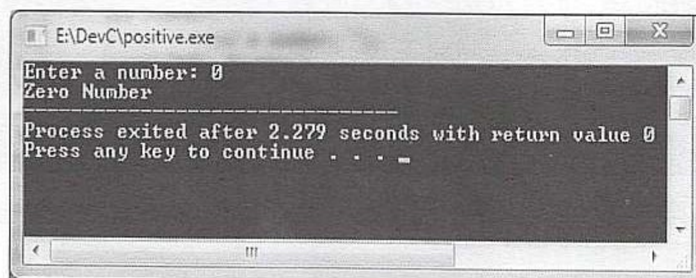
บรรทัดที่ 7 และ 8 นำค่า number มาเปรียบเทียบ ถ้า number = 0 แสดงข้อความ Zero Number

บรรทัดที่ 9 และ 10 ถ้า number > 0 แสดงข้อความ Positive Number

บรรทัดที่ 11 และ 12 ถ้าไม่ใช่ทั้ง 2 กรณีให้แสดงข้อความ Negative Number

ผลการรันโปรแกรม

โปรแกรมจะตรวจสอบว่าเป็นเลขจำนวนเต็มบวก ลบ หรือศูนย์ตามตัวเลขที่ป้อนเข้า



รูปที่ 2.9 ผลการรันโปรแกรมตรวจสอบเลขจำนวนเต็มบวก จำนวนเต็มลบ หรือศูนย์

ตัวอย่างที่ 2.7 โปรแกรมเปรียบเทียบหาค่ามากที่สุดของ 3 ตัวแปร

max=n1; n1 > n2 && n3

```
if(n1>n2 && n1>n3)
 max=n1;
```

max=n2; n2 > n1 && n3

```
else if(n2>n1 && n2>n3)
 max=n2;
```

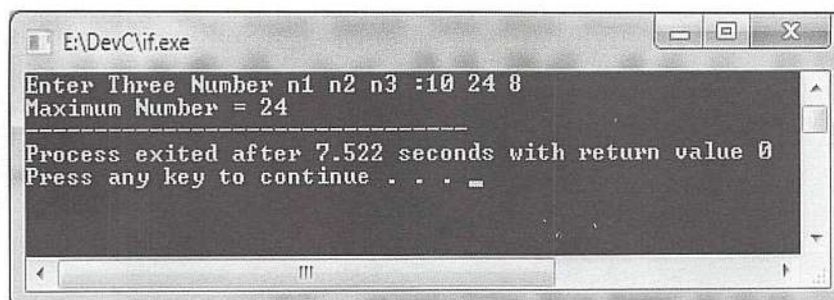
max=n3; n3

```
else
 max=n3;
```

```
#include <stdio.h>
int main()
 {
   int max,n1,n2,n3;
   printf("Enter Three Number n1 n2 n3 :");
   scanf("%d %d %d",&n1,&n2,&n3);
    if(n1>n2 && n1>n3)
      max=n1;
    else if(n2>n1 && n2>n3)
      max=n2;
    else
      max=n3;
   printf("Maximum Number = %d",max);
 }
```

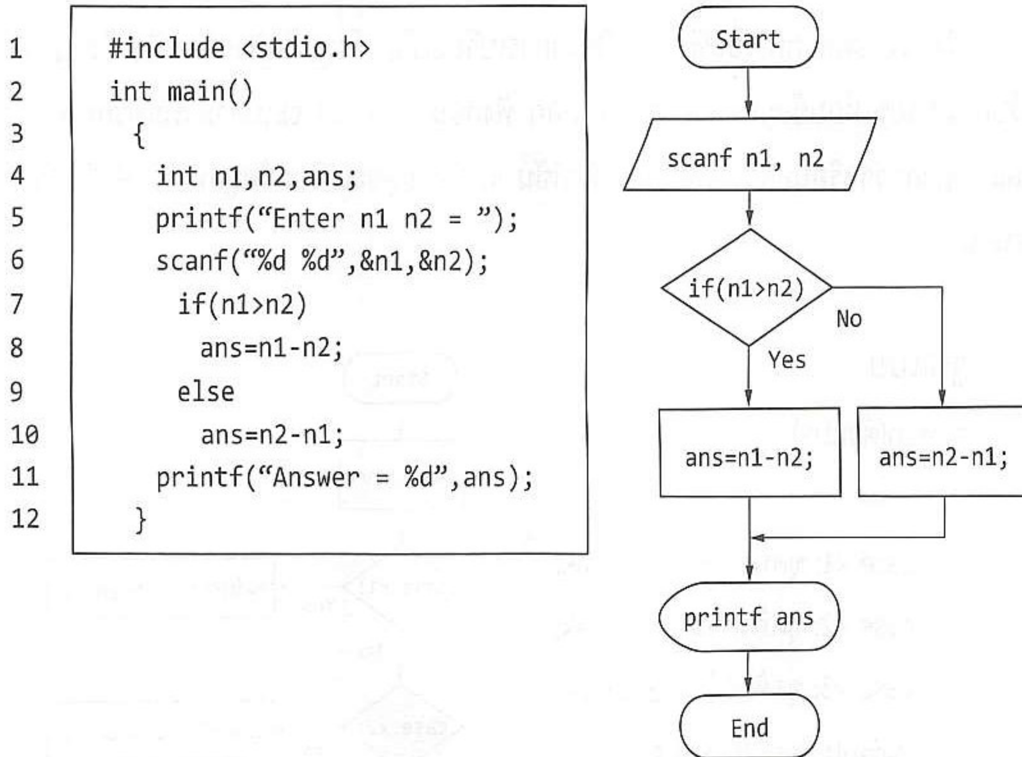
ผลการรันโปรแกรม

โปรแกรมจะเปรียบเทียบตัวแปร n1, n2 และ n3 นำค่ามากที่สุดมาเก็บไว้ในตัวแปร max แล้วแสดงค่ามากที่สุด



รูปที่ 2.10 ผลการรันโปรแกรมเปรียบเทียบหาค่ามากที่สุดของ 3 ตัวแปร

ตัวอย่างที่ 2.8 โปรแกรมหาผลต่างที่เป็นค่าบวกของเลข 2 จำนวน



```
1   #include <stdio.h>
2   int main()
3    {
4      int n1,n2,ans;
5      printf("Enter n1 n2 = ");
6      scanf("%d %d",&n1,&n2);
7        if(n1>n2)
8          ans=n1-n2;
9        else
10         ans=n2-n1;
11     printf("Answer = %d",ans);
12   }
```


รูปที่ 2.11 โฟลว์ชาร์ตของโปรแกรมหาผลต่างที่เป็นค่าบวกของเลข 2 จำนวน

คำอธิบาย

บรรทัดที่ 4 ประกาศ n1, n2 และ ans เป็นตัวแปรชนิดจำนวนเต็มขนาด 16 บิต

บรรทัดที่ 5 แสดงข้อความ Enter n1 n2 =

บรรทัดที่ 6 รอรับค่าตัวเลข 2 จำนวนมาเก็บไว้ในตัวแปร n1 และ n2

บรรทัดที่ 7 และ 8 เปรียบเทียบค่า n1, n2 ถ้า n1 > n2 ให้ ans = n1 - n2

บรรทัดที่ 9 และ 10 ถ้าไม่ใช่ (n2 > n1) ให้ ans = n2 - n1

ผลการรันโปรแกรม

โปรแกรมจะเปรียบเทียบค่า n1 และ n2 เพื่อหาจำนวนที่มากกว่า แล้วนำตัวเลขที่มีค่ามากกว่าลบด้วยตัวเลขที่น้อยกว่าก็จะเป็นคำตอบ

Enter n1 n2 = 8 12

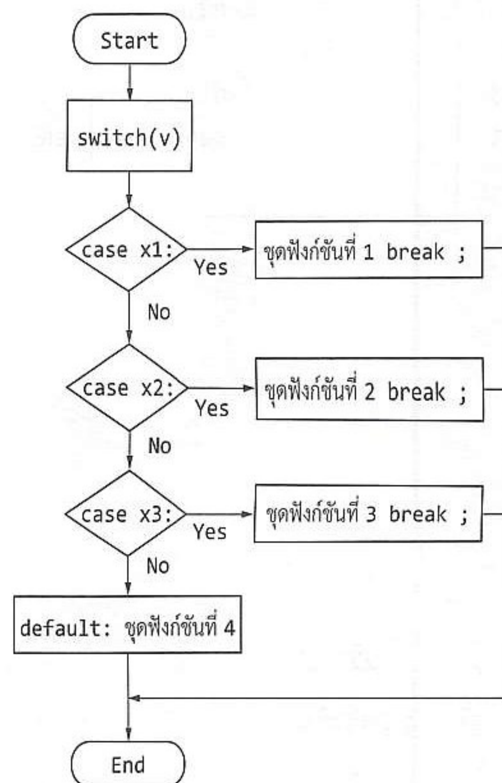
Answer = 4

2.6 ฟังก์ชัน switch()

ฟังก์ชัน switch() เป็นฟังก์ชันที่ใช้ในการเปรียบเทียบข้อมูลอีกฟังก์ชันหนึ่ง ซึ่งเหมาะสำหรับการเปรียบเทียบข้อมูลหลาย ๆ ทางเลือก ฟังก์ชัน switch() จะไม่สามารถเปรียบเทียบข้อมูลแบบมากกว่าหรือน้อยกว่าเหมือนกับฟังก์ชัน if() ได้ แต่จะเปรียบข้อมูลที่มีค่าเท่ากันหรือเหมือนกัน

รูปแบบ

```
switch(ตัวแปร)
  {
     case x1: ชุดฟังก์ชันที่ 1  break;
     case x2: ชุดฟังก์ชันที่ 2  break;
     case x3: ชุดฟังก์ชันที่ 3  break;
     default: ชุดฟังก์ชันที่ 4
  }
```



รูปที่ 2.12 โฟลว์ชาร์ตการทำงานของฟังก์ชัน switch()

การทำงานของฟังก์ชัน switch() จะนำตัวแปรมาเปรียบเทียบกับ case x1, x2, x3 ซึ่งถ้าตรงกับ case ใดหรือเงื่อนไขใด จะทำฟังก์ชันหลังกรณีนั้นหรือเงื่อนไขนั้นแล้วจบการทำงานของฟังก์ชัน switch() แต่ถ้าไม่ตรงกับกรณีใดหรือเงื่อนไขใดจะทำชุดฟังก์ชันที่อยู่หลัง default ซึ่งฟังก์ชัน default จะมีหรือไม่ก็ได้ และกรณีที่ใช้ในการเปรียบเทียบข้อมูลอาจมีมากกว่านี้ได้ ขึ้นอยู่กับลักษณะของการเขียนโปรแกรม ทั้งนี้ฟังก์ชัน switch() เหมาะสำหรับการเขียนโปรแกรมที่มีการเปรียบเทียบข้อมูลหลาย ๆ กรณีหรือหลาย ๆ เงื่อนไข

ตัวอย่างที่ 2.9 โปรแกรมตรวจสอบเลข 1 ถึง 5




```
1  #include <stdio.h>
2  int main()
3  {
4    int num;
5    printf("Enter a Number : ");
6    scanf("%d",&num);
7    switch(num)
8    {
9      case 1:   printf("One"); break;
10     case 2:   printf("Two"); break;
11     case 3:   printf("Three"); break;
12     case 4:   printf("Four"); break;
13     case 5:   printf("Five"); break;
14     default:  printf("Other Number");
15   }
16 }
```

คำอธิบาย

บรรทัดที่ 5 แสดงข้อความ Enter a Number :

บรรทัดที่ 6 รอรับค่าตัวเลขมาเก็บไว้ที่ตัวแปร num

บรรทัดที่ 7 ฟังก์ชัน switch() ทำการเปรียบเทียบตัวแปร num กับ case 1 ถึง case 5

บรรทัดที่ 9 ถึง 13 ถ้าค่าของตัวแปร num ตรงกับ case ใดจะทำคำสั่งใน case นั้น

บรรทัดที่ 14 แต่ถ้าไม่ตรงกับกรณีใดเลย จะทำคำสั่งหลัง default

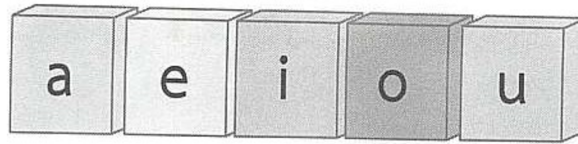
ผลการรันโปรแกรม

โปรแกรมจะตรวจสอบเลข 1 2 3 4 5 โดยรอรับค่าแล้วนำมาเปรียบเทียบ ถ้าตรงกับ case ใดก็ให้ทำคำสั่งใน case นั้น แต่ถ้าไม่ตรงกับเงื่อนไขใดให้ทำคำสั่งหลัง default



รูปที่ 2.13 ผลการรันโปรแกรมตรวจสอบตัวเลข 1 ถึง 5

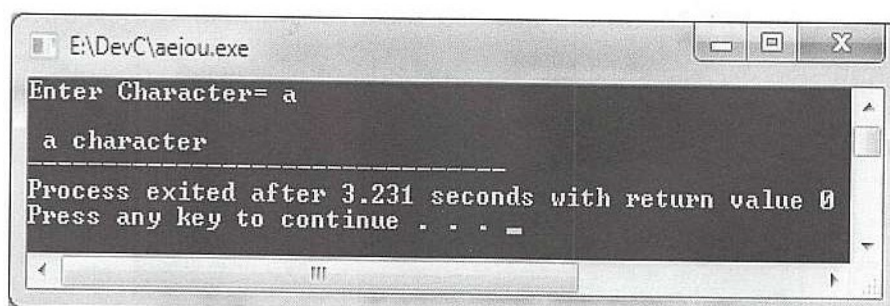
ตัวอย่างที่ 2.10 โปรแกรมตรวจสอบตัวอักษร a e i o u




```
#include <stdio.h>
int main()
 {
   char c;
   printf(“Enter Character= ”);
   scanf(“%c”,&c);
   switch(c)
   {
     case ‘a’: printf(“\n a character”);
               break;
     case ‘e’: printf(“\n e character”);
               break;
     case ‘i’: printf(“\n i character”);
               break;
     case ‘o’: printf(“\n o character”);
               break;
     case ‘u’: printf(“\n u character”);
               break;
    default : printf(“\n Other character”);
   }
 }
```

ผลการรันโปรแกรม

โปรแกรมจะตรวจสอบตัวอักษร a e i o u โดยรอรับค่าแล้วนำมาเปรียบเทียบ ถ้าตรงกับ case ใดก็ให้ทำใน case นั้น แต่ถ้าไม่ตรงกับเงื่อนไขใดให้ทำคำสั่งหลัง default



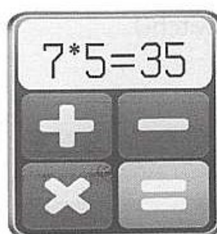
รูปที่ 2.14 ผลการรันโปรแกรมตรวจสอบตัวอักษร a e i o u

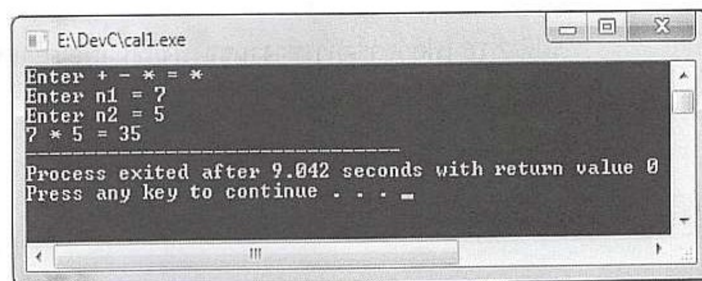
ตัวอย่างที่ 2.11 โปรแกรมบวก ลบ และคูณเลข 2 จำนวน

```
#include <stdio.h>
int main()
{
  char calcu;
  int n1,n2;
  printf(“Enter + - * = ”);
  scanf(“%c”,&calcu);
  printf(“Enter n1 = ”);
  scanf(“%d”,&n1);
  printf(“Enter n2 = ”);
  scanf(“%d”,&n2);
  switch(calcu)
   {
    case‘+’:
     printf(“%d + %d = %d”,n1,n2,n1+n2);
     break;
    case‘-’:
     printf(“%d - %d = %d”,n1,n2,n1-n2);
     break;
    case‘*’:
     printf(“%d * %d = %d”,n1,n2,n1*n2);
     break;
    default:
     printf(“Error! Calculate”);
   }
}
```

ผลการรันโปรแกรม

เมื่อป้อนเครื่องหมาย +, - หรือ * อย่างใดอย่างหนึ่งแล้วป้อนตัวเลข n1 และ n2 โปรแกรมจะนำค่า n1 และ n2 มาคำนวณตามเครื่องหมายที่ป้อน หากป้อนเครื่องหมายหรือสัญลักษณ์อื่นนอกเหนือจากนี้ โปรแกรมจะแสดงข้อความ Error! Calculate



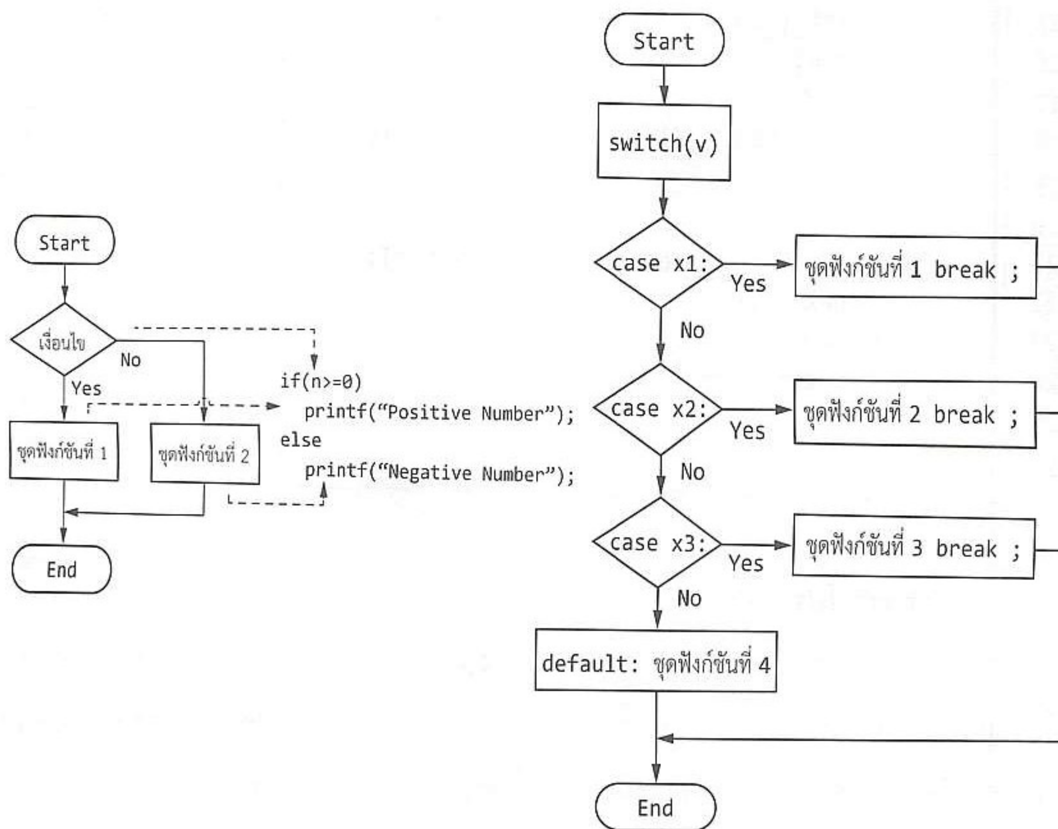


รูปที่ 2.15 ผลการรันโปรแกรมบวก ลบ และคูณเลข 2 จำนวน

2.7 สรุป

ฟังก์ชัน if() เป็นฟังก์ชันที่ใช้ในการเปรียบเทียบข้อมูล มีทั้งแบบทางเลือกเดียวและหลายทางเลือก ถ้าหากเงื่อนไขในการเปรียบเทียบเป็นจริงจะทำในชุดฟังก์ชันที่ 1 แต่ถ้าเงื่อนไขเป็นเท็จก็จะทำชุดฟังก์ชันที่ 2 ซึ่งคำสั่ง *if()* จะสามารถเปรียบเทียบค่าที่มากกว่า น้อยกว่า หรือเท่ากับได้

ฟังก์ชัน switch() เป็นฟังก์ชันที่ใช้ในการเปรียบเทียบข้อมูลอีกฟังก์ชันหนึ่ง ซึ่งเหมาะสำหรับการเปรียบเทียบข้อมูลหลาย ๆ ทางเลือก และฟังก์ชัน switch() ไม่สามารถเปรียบเทียบข้อมูลแบบมากกว่าหรือน้อยกว่าเหมือนกับฟังก์ชัน if() ได้ แต่จะเปรียบข้อมูลที่มีค่าเท่ากันหรือเหมือนกัน



รูปที่ 2.16 โฟลว์ชาร์ตการทำงานของฟังก์ชัน if else และ switch()

คำถามท้ายบทที่ 2

1. จงอธิบายการทำงานของฟังก์ชัน if()
2. จงอธิบายการทำงานของฟังก์ชัน switch()
3. ฟังก์ชัน if() กับฟังก์ชัน switch() ต่างกันอย่างไร
4. จงเขียนโปรแกรมตรวจสอบตัวอักษร a, e, i, o, u โดยใช้ฟังก์ชัน if()
5. จงเขียนโปรแกรมตรวจสอบตัวเลขที่หารด้วย 7 ลงตัว
6. จงเขียนโปรแกรมตรวจสอบหาตัวเลขมากที่สุดใน 3 จำนวน

3

การเขียนโปรแกรมวนรอบ

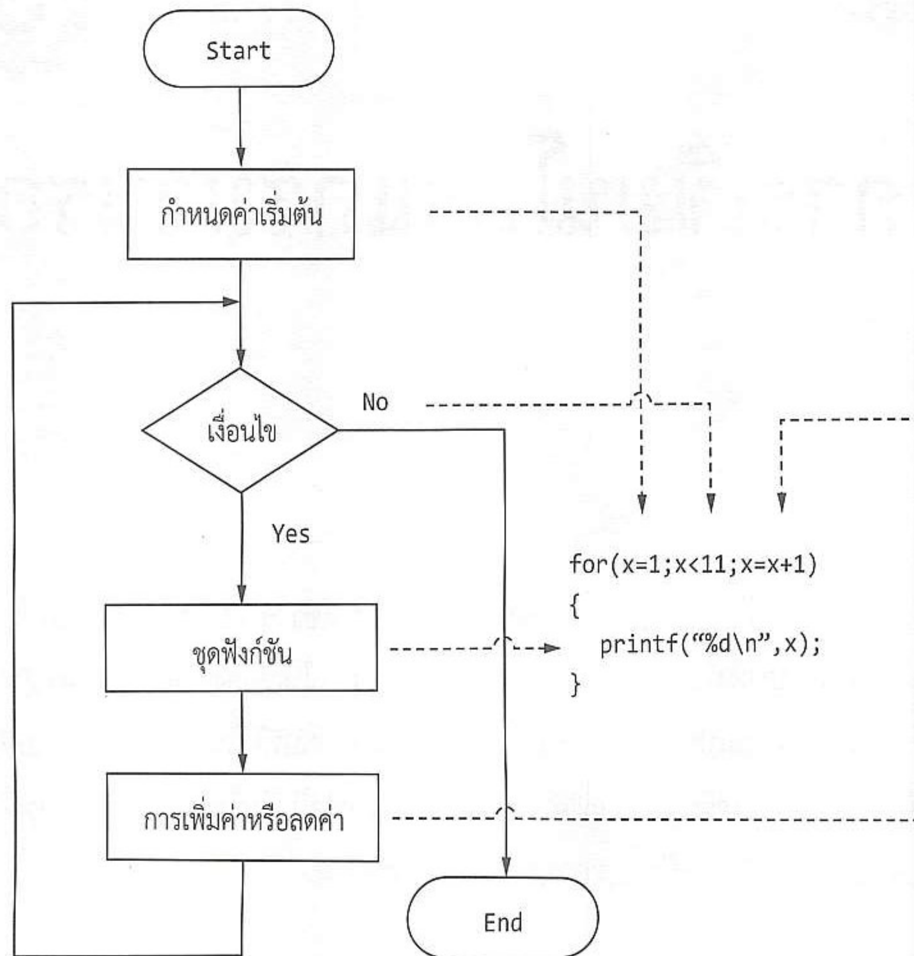
การวนรอบคือการทำงานที่มีลักษณะการทำคำสั่งที่ซ้ำ ๆ กัน สามารถใช้ฟังก์ชันวนลูป (Loop) หรือวนรอบการทำงานเพื่อช่วยลดขนาดของโปรแกรมให้เล็กลงและมีการทำงานที่รวดเร็วขึ้น ใช้เวลาในการเขียนโปรแกรมน้อยลง ซึ่งในภาษาซีมีฟังก์ชันที่ใช้ในการวนรอบหลายฟังก์ชัน ได้แก่ ฟังก์ชัน for() ฟังก์ชัน do_while(); และฟังก์ชัน while() ซึ่งเป็นฟังก์ชันวนรอบเหมือนกัน แต่มีรูปแบบของฟังก์ชันและลักษณะการทำงานที่ต่างกัน

3.1 ฟังก์ชัน for()

ฟังก์ชัน for() เป็นฟังก์ชันวนรอบ ใช้ในการเขียนโปรแกรมที่มีลักษณะการทำงานซ้ำ ๆ กัน การใช้งานฟังก์ชัน for() ส่วนใหญ่จะใช้กับการทำซ้ำที่รู้จำนวนรอบแน่นอน

รูปแบบ

```
for(ค่าเริ่มต้น;เงื่อนไข;การเพิ่มค่าหรือลดค่า)
   {
    ชุดฟังก์ชัน
   }
```



รูปที่ 3.1 การทำงานของฟังก์ชัน for()

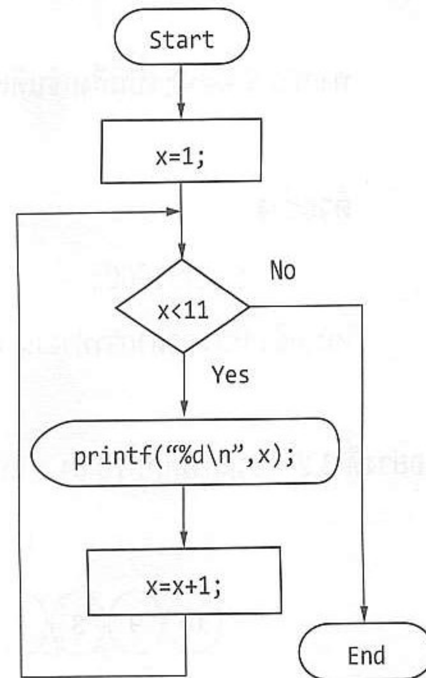
การทำงานของฟังก์ชัน for() ก่อนอื่นจะกำหนดค่าเริ่มต้นให้กับตัวแปร จากนั้นจึงตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริงจะทำชุดฟังก์ชันในปีกกาเปิดและปิด จากนั้นทำการเพิ่มค่าหรือลดค่าให้กับตัวแปรแล้วทำการตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริงจะทำต่อ แต่ถ้าเงื่อนไขเป็นเท็จจะออกจากลูป

ข้อควรระวัง

เครื่องหมายปีกกาเปิดและปิดในฟังก์ชัน for() ถ้าไม่ใส่ โปรแกรมจะทำฟังก์ชันเดียวเท่านั้น และฟังก์ชัน for() จะไม่มีเครื่องหมายเซมิโคลอนปิดท้าย ถ้าใส่เครื่องหมายเซมิโคลอนปิดท้ายฟังก์ชัน for() ในการรันและการคอมไพล์จะไม่เกิดข้อผิดพลาด (Error) แต่โปรแกรมจะไม่ทำตามเงื่อนไขที่กำหนด

ตัวอย่างที่ 3.1 โปรแกรมวนรอบนับ 1 ถึง 10 โดยใช้ฟังก์ชัน for()

```
1   #include <stdio.h>
2   int main()
3      {
4        char x;
5        for(x=1;x<11;x=x+1)
6          printf("Loop = %d\n",x);
7      }
```



รูปที่ 3.2 โฟลว์ชาร์ตการทำงานของโปรแกรมนับ 1 ถึง 10

คำอธิบาย

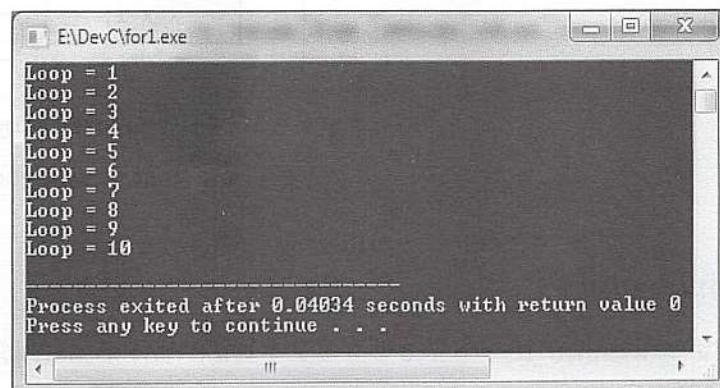
บรรทัดที่ 4 ประกาศตัวแปร x เป็นชนิดตัวอักษรหรือจำนวนเต็มขนาด 8 บิต

บรรทัดที่ 5 ฟังก์ชัน for() ทำการวนรอบ 10 ครั้ง โดยค่าเริ่มต้น x = 1 เงื่อนไขในการวนรอบ x < 11 และทำการเพิ่มค่าตัวแปรทีละหนึ่ง (x = x + 1)

บรรทัดที่ 6 ถ้าเงื่อนไขเป็นจริงให้พิมพ์ข้อความ Loop = 1 ถึง Loop = 10 ตามค่าตัวแปร x

ผลการรันโปรแกรม

โปรแกรมจะวนรอบนับ 1 ถึง 10 และแสดงข้อความดังนี้



รูปที่ 3.3 ผลการรันโปรแกรมวนรอบนับ 1 ถึง 10

3.2 ฟังก์ชัน Sleep();

ฟังก์ชัน Sleep(); เป็นฟังก์ชันที่ใช้ในการหน่วงเวลา ซึ่งถูกนิยามไว้ใน windows.h

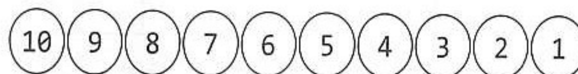
ตัวอย่าง

Sleep(1000);

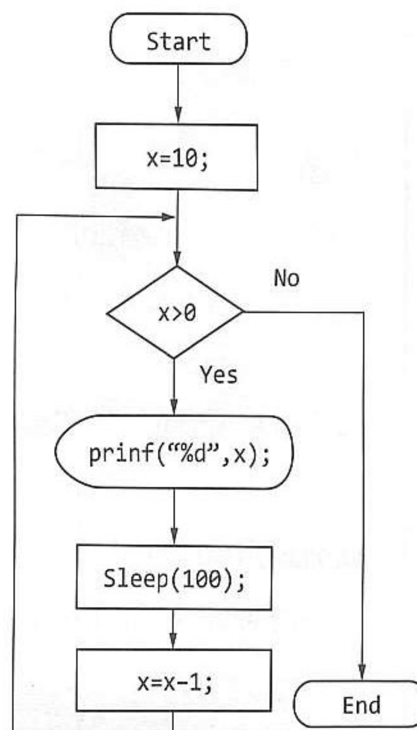
หมายถึง หน่วงเวลาการทำงาน 1000 มิลลิวินาที (ms)

ตัวอย่างที่ 3.2 โปรแกรมนับ 10 ถึง 1 แบบหน่วงเวลา

Loop x = 10-1




```
#include <stdio.h>
#include <windows.h>
int main()
{
  int x;
  for(x=10;x>0;x--)
    { printf(“%d ”,x);
      Sleep(100);
    }
}
```



รูปที่ 3.4 โฟลว์ชาร์ตการทำงานของโปรแกรม นับ 10 ถึง 1 แบบหน่วงเวลา

ผลการรันโปรแกรม

โปรแกรมจะวนรอบนับ 10 9 8 7 6 5 4 3 2 1 ตามค่าของ x โดยค่าเริ่มต้น x = 10 เงื่อนไข x > 0 และลดค่าของ x ทีละหนึ่ง ควรระวังตรงเงื่อนไข x> หรือ x< ซึ่งมักผิดบ่อย

ตัวอย่างที่ 3.3 โปรแกรมบวกเลข 1 ถึง 8

1+2+3+4+5+6+7+8 = ?

```
1   #include <stdio.h>
2   int main()
3   {
4    int x,n=8,sum=0;
5      for(x=1;x<=n;x++)
6      {
7        sum=sum+x;
8        printf("STEP %d = %d\n",x,sum);
9      }
10  }
```

คำอธิบาย

บรรทัดที่ 4 ประกาศตัวแปรและกำหนดค่าเริ่มต้นให้ตัวแปร (int x,n=8,sum=0;)

บรรทัดที่ 5 for() วนรอบเพื่อบวกเลข 8 รอบ x = 1 ถึง 8

บรรทัดที่ 7 บวกค่าในตัวแปร sum=sum+x เป็นการบวกเลข 1+2+3+4+5+6+7+8=36

บรรทัดที่ 8 แสดงผลการบวกในแต่ละรอบ

ผลการรันโปรแกรม

โปรแกรมจะวนรอบบวกเลข 1 ถึง 8 จากตัวแปร x และ sum แล้วแสดงผลการบวก

```
E:\DevC\sumn.exe
STEP 1 = 1
STEP 2 = 3
STEP 3 = 6
STEP 4 = 10
STEP 5 = 15
STEP 6 = 21
STEP 7 = 28
STEP 8 = 36

--------------------------------
Process exited after 0.07124 seconds with return value 0
Press any key to continue . . .
```

รูปที่ 3.5 ผลการรันโปรแกรมการบวกเลข 1 ถึง 8

ตัวอย่างที่ 3.4 โปรแกรมสูตรคูณ

Multiplication Table

6	7	8
6*1 =6	7*1 =7	8*1 =8
6*2=12	7*2=14	8*2=16
6*3=18	7*3=21	8*3=24
6*4=24	7*4=28	8*4=32
6*5=30	7*5=35	8*5=40
6*6=36	7*6=42	8*6=48
6*7=42	7*7=49	8*7=56
6*8=48	7*8=56	8*8=64
6*9=54	7*9=63	8*9=72

```
1   #include <stdio.h>
2   int main()
3   {
4      int n,x;
5      printf("Enter an integer = ");
6      scanf("%d",&n);
7      for(x=1;x<10;x++)
8      {
9        printf("%d * %d = %d\n",n,x,n*x);
10     }
11  }
```

คำอธิบาย

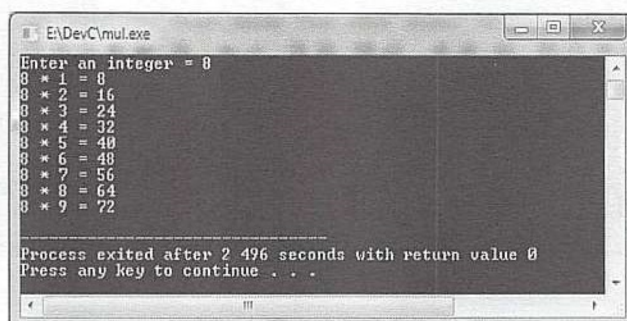
บรรทัดที่ 4 ประกาศตัวแปร n, x เป็นชนิดจำนวนเต็ม

บรรทัดที่ 6 รอรับค่าตัวเลขแม่สูตรคูณมาเก็บไว้ที่ตัวแปร n

บรรทัดที่ 7 ถึง 10 วนรอบ 9 ครั้ง แสดงสูตรคูณตามค่าตัวแปร x

ผลการรันโปรแกรม

โปรแกรมจะแสดงสูตรคูณตามตัวเลขแม่สูตรคูณที่ป้อน



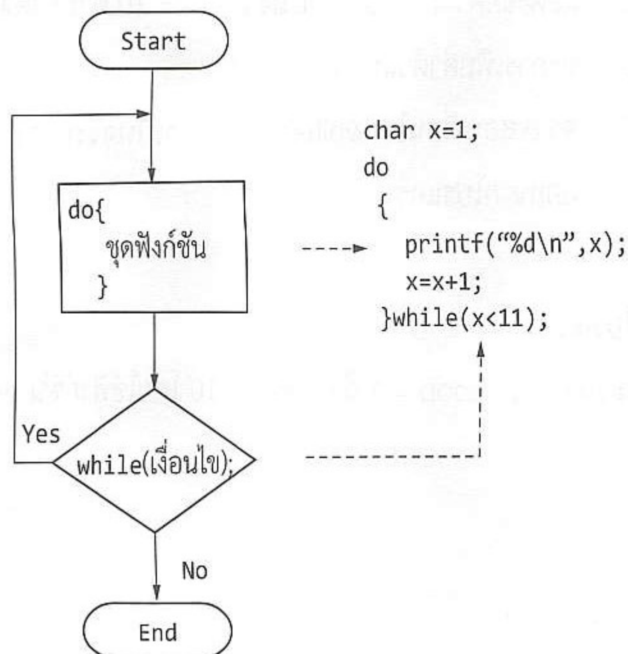
รูปที่ 3.6 ผลการรันโปรแกรมสูตรคูณ

3.3 ฟังก์ชัน do_while();

ฟังก์ชัน do_while(); เป็นฟังก์ชันวนรอบ ทำงานในลักษณะที่ซ้ำกัน โดยการทำงานจะทำชุดฟังก์ชันในปีกกาเปิดและปิดก่อน จากนั้นจะทำการตรวจสอบเงื่อนไข ถ้าเงื่อนไขเป็นจริงก็จะทำชุดฟังก์ชันต่อ แต่ถ้าเงื่อนไขเป็นเท็จจะออกจากการวนรอบ

รูปแบบ

```
do
  {
   ชุดฟังก์ชัน
  }whlie(เงื่อนไข);
```



รูปที่ 3.7 โฟลว์ชาร์ตการทำงานของฟังก์ชัน do_while();

ตัวอย่างที่ 3.5 โปรแกรมวนรอบนับ 1 ถึง 10 โดยใช้ฟังก์ชัน do_while();

```
1  #include <stdio.h>
2  int main()
3     { char x=1;
4       do{
5          printf("Loop = %d\n",x);
6          x=x+1;
7         }while(x<11);
8     }
```

คำอธิบาย

บรรทัดที่ 3 กำหนดค่าเริ่มต้นให้ตัวแปร x = 1

บรรทัดที่ 4 ทำคำสั่งใน do{ }

บรรทัดที่ 5 แสดงข้อความ Loop = 1 ถึง Loop = 10 ตามค่าตัวแปร x

บรรทัดที่ 6 ทำการเพิ่มค่าตัวแปร x = x + 1

บรรทัดที่ 7 ตรวจสอบเงื่อนไข while(x<11); ถ้าเป็นจริงทำงานต่อ แต่ถ้าเป็นเท็จออกจากโปรแกรม

ผลการรันโปรแกรม

โปรแกรมจะวนรอบนับ Loop = 1 ถึง Loop = 10 โดยใช้ฟังก์ชัน do_while();

ตัวอย่างที่ 3.6 โปรแกรมแปลงเลขฐาน 10 เป็นเลขฐาน 2

```
                  เศษ
2 | 11             1  ↑
   2 | 5           1  |
      2 | 2        0  |
         2 | 1     1  |
             0
```

11 = 1011 BIN

```
1   #include <stdio.h>
2   int main()
3   {
4      int num;
5      printf("Enter Number : ");
6      scanf("%d",&num);
7      do
8        {
9         printf("%d\n",num%2);
10           num=num/2;
11       }while(num>0);
12  }
```

คำอธิบาย

บรรทัดที่ 6 รอรับค่าตัวเลขมาเก็บไว้ที่ตัวแปร num

บรรทัดที่ 7 ทำคำสั่งใน do{ }

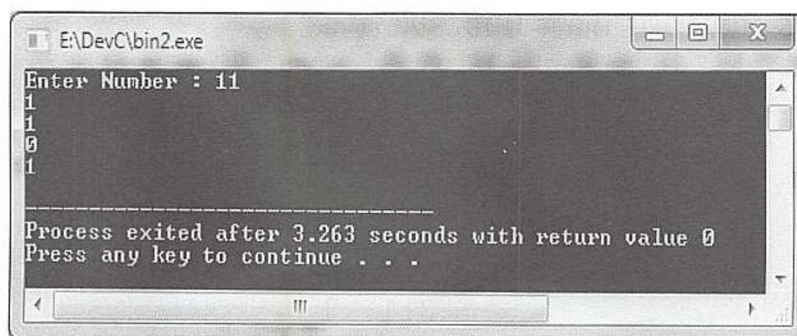
บรรทัดที่ 9 แสดงเศษของการหารด้วย 2 (num%2) ซึ่งจะได้เศษ 0 หรือ 1 เท่านั้น

บรรทัดที่ 10 หารค่าด้วย 2 (num = num/2)

บรรทัดที่ 11 ตรวจสอบเงื่อนไข while(num>0); วนรอบหารจนกว่า num = 0

ผลการรันโปรแกรม

โปรแกรมจะรอรับค่าตัวเลขแล้ววนรอบหารจนกว่าค่าของตัวแปร num เป็นศูนย์ โดยคำตอบที่ได้จะต้องกลับค่าจากล่างขึ้นบน คำตอบคือ 1 0 1 1 เลขฐานสอง ซึ่งมีค่าเท่ากับ 11



รูปที่ 3.8 ผลการรันโปรแกรมแปลงเลขฐาน 10 เป็นเลขฐาน 2

3.4 ฟังก์ชัน while()

ฟังก์ชัน while() เป็นฟังก์ชันวนรอบซ้ำ การทำงานจะทำการตรวจสอบเงื่อนไขก่อน ถ้าเงื่อนไขเป็นจริงจะทำชุดฟังก์ชันภายในปีกกาเปิดและปิด แต่ถ้าเงื่อนไขเป็นเท็จจะออกจากฟังก์ชัน while()

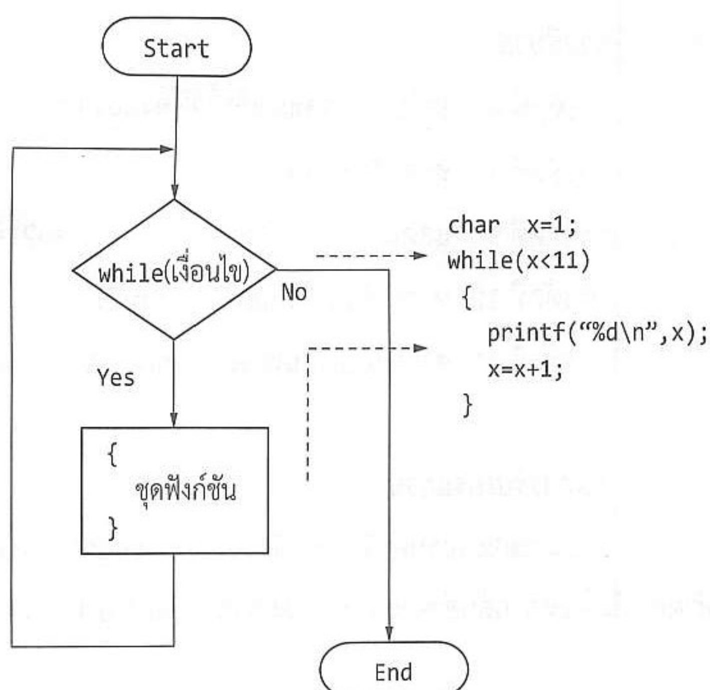
รูปแบบ

while(เงื่อนไข)

{

ชุดฟังก์ชัน

}



รูปที่ 3.9 โฟลว์ชาร์ตการทำงานของฟังก์ชัน while()

ตัวอย่างที่ 3.7 โปรแกรมวนรอบนับ 1 ถึง 10 โดยใช้ฟังก์ชัน while()

```
1  #include <stdio.h>
2  int main()
3     { char x=1;
4       while(x<11)
5       {
6         printf("Loop = %d\n",x);
7         x=x+1;
8       }
9     }
```

คำอธิบาย

บรรทัดที่ 3 ประกาศตัวแปร x และกำหนดค่าเริ่มต้นให้เท่ากับ 1

บรรทัดที่ 4 ตรวจสอบเงื่อนไข while(x<11) ถ้าเป็นจริงทำคำสั่งในปีกกาเปิดและปิด แต่ถ้าเงื่อนไขเป็นเท็จจะออกจากฟังก์ชัน while()

บรรทัดที่ 6 แสดงข้อความ Loop = 1 ถึง Loop = 10 ตามค่าตัวแปร x

บรรทัดที่ 7 ทำการเพิ่มค่าตัวแปร x = x + 1 แล้ววนรอบกลับไปตรวจสอบเงื่อนไข while(x<11)

ผลการรันโปรแกรม

โปรแกรมจะวน 10 รอบ แสดงข้อความ Loop = 1 ถึง Loop = 10 โดยใช้ฟังก์ชัน while()

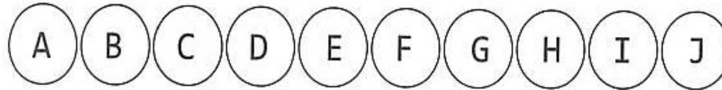
```
E:\DevC\while1-10.exe
Loop = 1
Loop = 2
Loop = 3
Loop = 4
Loop = 5
Loop = 6
Loop = 7
Loop = 8
Loop = 9
Loop = 10

--------------------------------
Process exited after 0.08251 seconds with return value 0
Press any key to continue . . .
```

รูปที่ 3.10 ผลการรันโปรแกรมวนรอบนับ 1 ถึง 10 โดยใช้ฟังก์ชัน while()

ตัวอย่างที่ 3.8 โปรแกรมวนรอบพิมพ์ตัวอักษร A-J

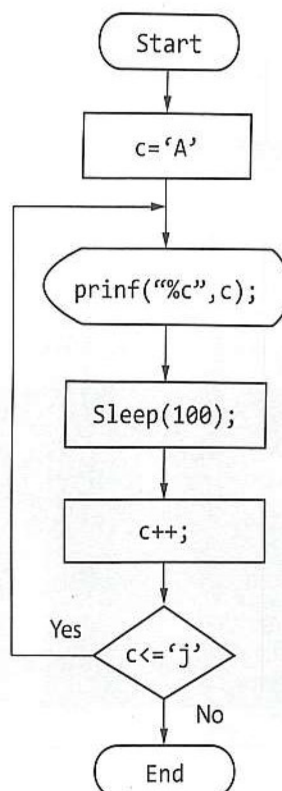
Loop C = ‘A’–‘J’




```
#include <stdio.h>
#include <windows.h>
int main()
{
   char c=‘A’;
    do{
       printf(“ %c”,c);
       Sleep(100);
       c++;
     }while(c<=‘J’);
}
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบพิมพ์ตัวอักษร A B C D E F G H I J



รูปที่ 3.11 โฟลว์ชาร์ตแสดงการทำงานของโปรแกรมพิมพ์ตัวอักษร A-J

ตัวอย่างที่ 3.9 โปรแกรมหาจำนวนหลัก

```
1   #include <stdio.h>
2   int main()
3   {
4    int num,count=0;
5    printf("Enter a Number :");
6    scanf("%d",&num);
7     while(num!=0)
8     {
9      num = num/10;
10      count++;
11     printf("%d %d\n",num,count);
12    }
13   printf("Number of digits: %d",count);
14  }
```

คำอธิบาย

บรรทัดที่ 6 รอรับค่าตัวเลขมาเก็บไว้ที่ตัวแปร num

บรรทัดที่ 7 วนรอบคำสั่งถ้าตัวแปร num ไม่เท่ากับศูนย์

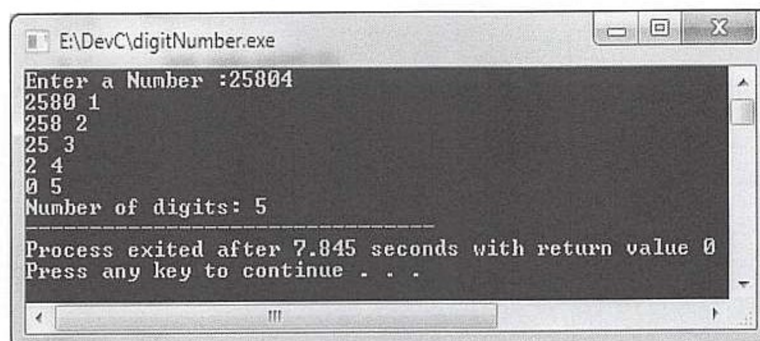
บรรทัดที่ 9 ลดค่าตัวแปร num ทีละหลัก เช่น 5421/10 = 542

บรรทัดที่ 10 นับหาจำนวนหลัก (count++;)

บรรทัดที่ 11 แสดงตัวเลขและจำนวนหลักในการนับแต่ละรอบ

ผลการรันโปรแกรม

โปรแกรมจะรอรับค่าตัวเลขแล้ววนรอบหารด้วย 10 ซึ่งเป็นการลดหลักของตัวแปร num และนับจำนวนหลักโดยใช้คำสั่ง count++;



รูปที่ 3.12 ผลการรันโปรแกรมหาจำนวนหลัก

ตัวอย่างที่ 3.10 โปรแกรมหาค่าตัวหารร่วมมาก

```
1   #include <stdio.h>
2   int main()
3   {
4     int n1,n2;
5     printf("Enter two positive integers: ");
6     scanf("%d %d",&n1,&n2);
7     while(n1!=n2)
8     {
9        if(n1 > n2)
10           n1 = n1-n2;
11       else
12           n2 = n2-n1;
13       printf("%d %d\n",n1,n2);
14    }
15    printf("GCD = %d",n1);
16  }
```

คำอธิบาย

บรรทัดที่ 6 รอรับค่าตัวเลขสองค่ามาเก็บที่ตัวแปร n1 และ n2

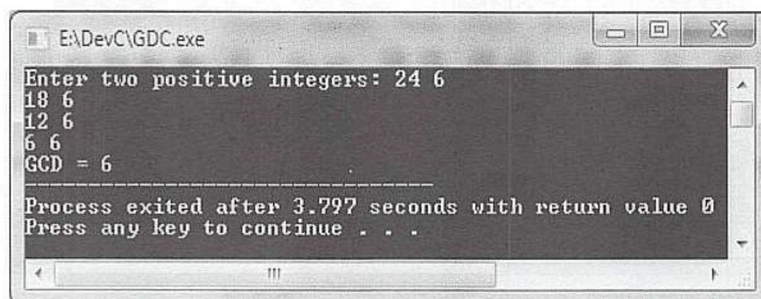
บรรทัดที่ 7 วนรอบคำสั่งถ้าตัวแปร n1 ≠ n2 เมื่อ n1 = n2 แสดงว่าได้ค่าตัวหารร่วมมาก

บรรทัดที่ 9 และ 10 ถ้า n1 > n2 ให้ n1 = n1 - n2

บรรทัดที่ 11 และ 12 ถ้าไม่ใช่ ให้ n2 = n2 - n1

ผลการรันโปรแกรม

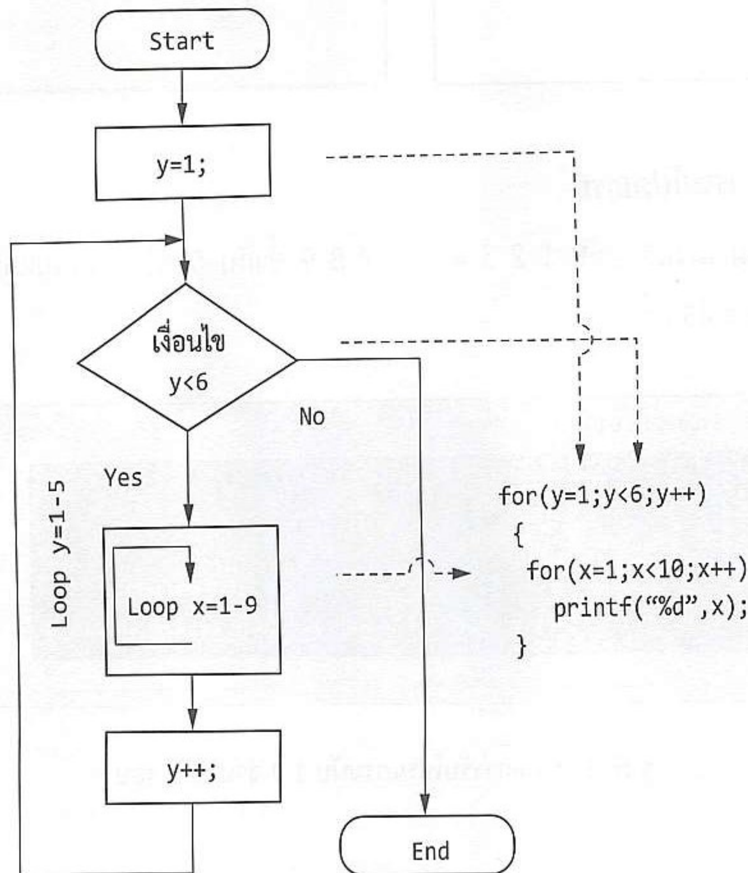
โปรแกรมจะรอรับค่าตัวเลข n1 และ n2 จากนั้นเปรียบเทียบค่า n1 กับ n2 เพื่อหาค่าตัวเลขที่มากกว่าแล้วลบด้วยตัวเลขที่น้อยกว่า โดยจะวนรอบซ้ำจนกว่า n1 = n2 ซึ่งเป็นค่าของตัวหารร่วมมาก



รูปที่ 3.13 ผลการรันโปรแกรมหาค่าตัวหารร่วมมาก

3.5 การวนรอบซ้ำซ้อน

การวนรอบซ้ำซ้อน หรือลูปซ้อนลูป ซึ่งจะต้องมีการใช้ฟังก์ชันวนรอบซ้อนกัน เพื่อใช้ในการเขียนโปรแกรมที่ซับซ้อน เช่น การเรียงข้อมูล การค้นหาข้อมูล การควบคุมที่เป็นลักษณะสองมิติ หรือการวนรอบนับของนาฬิกาที่จะมีวงรอบวินาที นาที และชั่วโมง การวนซ้ำอาจใช้ฟังก์ชัน for() ฟังก์ชัน while() และฟังก์ชัน do_while(); เพียงอย่างเดียวหรือจะใช้ร่วมกันก็ได้ จากรูปที่ 3.14 การวนรอบในลูป x ซึ่งเป็นลูปด้านในจะนับ 1 ถึง 9 ส่วนลูป y จะทำการวนรอบ 5 ครั้ง ซึ่งทั้งหมดจะวนรอบ 45 ครั้ง



รูปที่ 3.14 โฟลว์ชาร์ตการทำงานการวนรอบซ้ำซ้อน

ตัวอย่างที่ 3.11 โปรแกรมนับ 1-9 จำนวน 5 รอบ

```
#include <stdio.h>
int main()
{ int x,y;
  for(y=1;y<6;y++)
  {
    for(x=1;x<10;x++)
        printf("%d ",x);
    printf("\n");
  }
}
```

```
// ประกาศตัวแปร x, y เป็นชนิดจำนวนเต็ม
// y วนรอบ 1-5

// x วนรอบ 1-9
// พิมพ์ค่า x 1-9
// ขึ้นบรรทัดใหม่
```

ผลการรันโปรแกรม

โปรแกรมวนรอบนับ 1 2 3 4 5 6 7 8 9 ซ้ำกัน 5 ครั้ง รวมกันแล้วโปรแกรมจะวนรอบ 5 x 9 = 45 ครั้ง

```
E:\DevC\for1-10.exe
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9

--------------------------------
Process exited after 0.03361 seconds with return value 0
Press any key to continue . . .
```

รูปที่ 3.15 ผลการรันโปรแกรมนับ 1-9 จำนวน 5 รอบ

ตัวอย่างที่ 3.12 โปรแกรมนับ 1-5 แบบเพิ่มจำนวน

```
#include <stdio.h>
int main()
{ int x,y;
  for(y=1;y<6;y++)
   {
     for(x=1;x<=y;x++)
       printf("%d ",x);
     printf("\n");
   }
}
```

```
// คำอธิบาย

// ประกาศตัวแปร x, y เป็นชนิดจำนวนเต็ม
// y วนรอบ 1-5

// x วนรอบ 1-y
// พิมพ์ค่า x
// ขึ้นบรรทัดใหม่
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบพิมพ์ตัวเลขดังนี้

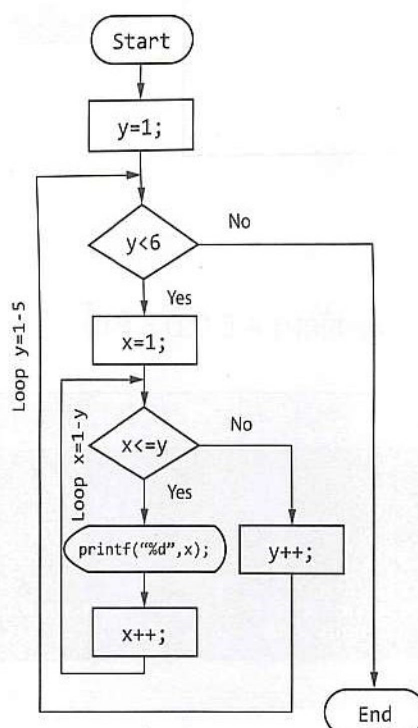
1

1 2

1 2 3

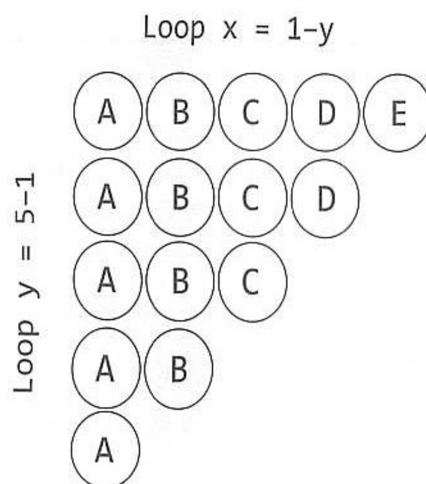
1 2 3 4

1 2 3 4 5



รูปที่ 3.16 โฟลว์ชาร์ตการทำงานของโปรแกรมนับ 1-5 แบบเพิ่มจำนวน

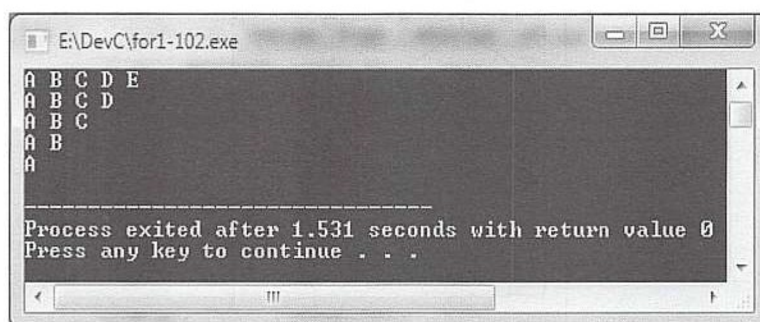
ตัวอย่างที่ 3.13 โปรแกรมพิมพ์ตัวอักษร A B C D E แบบลดจำนวน




```
#include <stdio.h>
#include <windows.h>
int main()
{ int x,y;
  for(y=5;y>=1;y--)
  {
   for(x='A';x<y+'A';x++)
   {
    printf("%c ",x);
    Sleep(100);
   }
   printf("\n");
  }
}
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบพิมพ์ตัวอักษร A B C D E ดังนี้

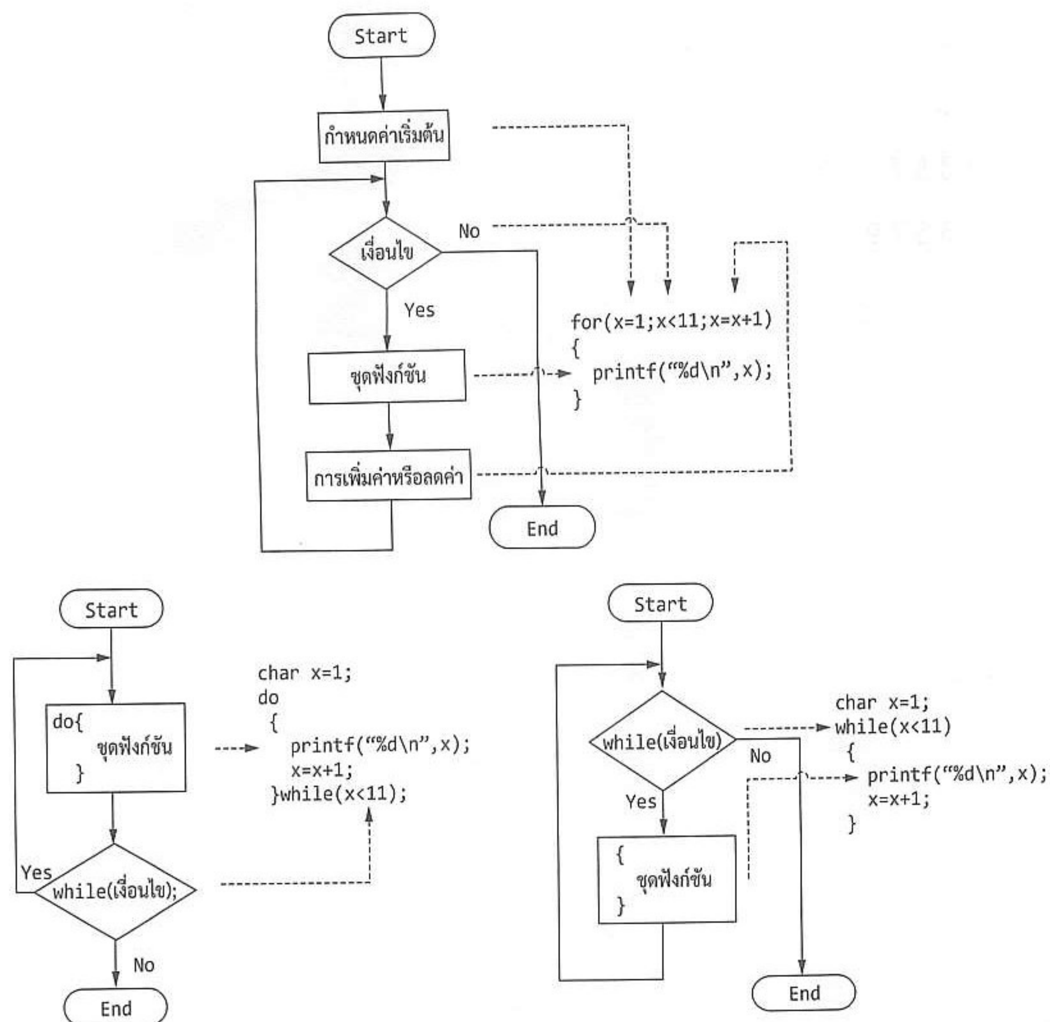


รูปที่ 3.17 ผลการรันโปรแกรมพิมพ์ตัวอักษร A B C D E แบบลดจำนวน

3.6 สรุป

การเขียนโปรแกรมที่มีลักษณะการทำงานวนรอบซ้ำ ๆ กัน มีฟังก์ชันหลักที่ใช้กันทั่วไป ได้แก่ ฟังก์ชัน for() ฟังก์ชัน do_while(); และ while() ซึ่งการใช้งานฟังก์ชัน for() ส่วนใหญ่จะใช้กับการวนรอบซ้ำที่รู้จำนวนรอบแน่นอน โดยจะกำหนดค่าเริ่มต้นให้กับตัวแปรก่อน จากนั้นตรวจสอบเงื่อนไข ถ้าเป็นจริงจะทำชุดฟังก์ชันในปีกกาเปิดและปิด ถัดไปจึงเพิ่มหรือลดค่าให้กับตัวแปรแล้วกลับมาตรวจสอบเงื่อนไขอีกครั้ง ซึ่งการทำงานจะวนลูปเช่นนี้ไปเรื่อย ๆ จนกว่าเงื่อนไขจะเป็นเท็จและออกจากลูป

ส่วนฟังก์ชัน do_while(); และฟังก์ชัน while() ก็เป็นการทำงานวนรอบซ้ำเช่นกัน แต่ฟังก์ชัน do_while(); จะทำชุดฟังก์ชันในปีกกาเปิดและปิดก่อนจากนั้นจึงตรวจสอบเงื่อนไข ส่วนฟังก์ชัน while() จะตรวจสอบเงื่อนไขก่อน ถ้าเป็นจริงจะทำชุดฟังก์ชันในปีกกาเปิดและปิด แต่ถ้าเป็นเท็จจะออกจากฟังก์ชัน



รูปที่ 3.18 การทำงานของฟังก์ชัน for() ฟังก์ชัน do_while(); และ while()

คำถามท้ายบทที่ 3

1. จงอธิบายการทำงานของฟังก์ชัน for()
2. จงอธิบายการทำงานของฟังก์ชัน do_while();
3. จงอธิบายการทำงานของฟังก์ชัน while()
4. ฟังก์ชัน while() และ do_while(); ต่างกันอย่างไร
5. จงอธิบายการทำงานของฟังก์ชัน Sleep();
6. จงเขียนโปรแกรมหาค่าตัวคูณร่วมน้อยของเลข 2 จำนวน
7. จงเขียนโฟลว์ชาร์ตของโปรแกรมหาค่าตัวคูณร่วมน้อย
8. จงเขียนโปรแกรมพิมพ์ตัวเลขดังนี้

```
1
1 3
1 3 5
1 3 5 7
1 3 5 7 9
```



อาร์เรย์และฟังก์ชัน

ในการเขียนโปรแกรมภาษาซีต้องมีการประกาศตัวแปรก่อนเสมอ เมื่อต้องการนำตัวแปรนั้นมาใช้งาน ในบางโปรแกรมที่จำเป็นต้องใช้ตัวแปรหลาย ๆ ตัวเพื่อการคำนวณหรือการเก็บข้อมูล การประกาศตัวแปรแบบธรรมดานั้นไม่ค่อยสะดวกนัก ในภาษาซีจึงมีตัวแปรแบบอาร์เรย์หรือตัวแปรชุดเพื่ออำนวยความสะดวกในการเขียนโปรแกรม

4.1 อาร์เรย์

อาร์เรย์ (Array) คือตัวแปรแถวลำดับหรือตัวแปรชุดที่เป็นชนิดเดียวกัน ในการประกาศตัวแปรแบบอาร์เรย์จะมีเครื่องหมาย [ ] ต่อท้ายเพื่อบอกขนาดของตัวแปรอาร์เรย์ ซึ่งตัวแปรอาร์เรย์มีขนาดตั้งแต่ 1 มิติ 2 มิติ และ 3 มิติ โดยมีรายละเอียดดังนี้

อาร์เรย์ 1 มิติ คือตัวแปรชุดซึ่งเป็นตัวแปรชนิดเดียวกันที่มี 1 มิติ

อาร์เรย์ 2 มิติ คือตัวแปรชุดซึ่งเป็นตัวแปรชนิดเดียวกันที่มี 2 มิติ

อาร์เรย์ 3 มิติ คือตัวแปรชุดซึ่งเป็นตัวแปรชนิดเดียวกันที่มี 3 มิติ

ตัวอย่างการกำหนดตัวแปรอาร์เรย์แบบ 1 มิติ

```
char sw[5];
```

หมายถึง การประกาศตัวแปร sw ให้เป็นตัวแปรอาร์เรย์มีขนาดเท่ากับ 5 หน่วย ประกอบไปด้วยตัวแปรอาร์เรย์ sw[0] ถึง sw[4] แต่ละตัวจะเก็บข้อมูลได้ 1 ไบต์ แสดงดังรูปที่ 4.1

sw[0]	sw[1]	sw[2]	sw[3]	sw[4]
1 Byte	1 Byte	1 Byte	1 Byte	1 Byte

รูปที่ 4.1 การประกาศตัวแปรอาร์เรย์ sw[5]

int ar[10];

หมายถึง การประกาศตัวแปร ar ให้เป็นตัวแปรแบบอาร์เรย์มีขนาดเท่ากับ 10 หน่วย ประกอบไปด้วยตัวแปรอาร์เรย์ ar[0] ถึง ar[9] แต่ละตัวจะเก็บข้อมูลได้ 2 ไบต์ แสดงดังรูปที่ 4.2

ar[0]	ar[1]	ar[2]	ar[3]	ar[4]	ar[5]	ar[6]	ar[7]	ar[8]	ar[9]
2 Byte	2 Byte	2 Byte	2 Byte	2 Byte	2 Byte	2 Byte	2 Byte	2 Byte	2 Byte

รูปที่ 4.2 การประกาศตัวแปรอาร์เรย์ ar[10]

ตัวอย่างที่ 4.1 โปรแกรมกำหนดค่า 0 ถึง 4 ให้กับตัวแปรอาร์เรย์ 1 มิติ

```
#include <stdio.h>
int main()
{ int A[5],x;
    for(x=0;x<5;x++)
     { A[x]=x;
       printf("A[%d]=%d",x,A[x]);
     }
}
```

ผลการรันโปรแกรม

โปรแกรมจะกำหนดค่า 0 ถึง 4 ให้กับตัวแปรอาร์เรย์ A[0] ถึง A[4] ซึ่งเป็นตัวแปรอาร์เรย์ 1 มิติดังนี้ A[0] = 0, A[1] = 1, A[2] = 2, A[3] = 3 และ A[4] = 4

A[0]	A[1]	A[2]	A[3]	A[4]
0	1	2	3	4

ตัวอย่างการกำหนดตัวแปรอาร์เรย์แบบ 2 มิติ

int A2[2][5];

หมายถึง การประกาศตัวแปร A2 ให้เป็นตัวแปรอาร์เรย์ขนาด 2 มิติ แต่ละหน่วยเก็บข้อมูลได้ 16 บิต ซึ่งประกอบด้วยตัวแปรอาร์เรย์ A2[0][0] ถึง A2[1][4] จำนวนทั้งหมด 10 หน่วยดังนี้

	[0]	[1]	[2]	[3]	[4]
A2[0][ ]					
A2[1][ ]					

รูปที่ 4.3 ตัวแปรอาร์เรย์ขนาด 2 มิติ

ตัวอย่างที่ 4.2 โปรแกรมกำหนดค่า 0 ถึง 9 ให้กับตัวแปรอาร์เรย์ 2 มิติ

```
#include <stdio.h>
int main()
{ int A2[2][5],x,y,data=0;
   for(x=0;x<2;x++)
    for(y=0;y<5;y++)
     { A2[x][y]=data;
       data=data+1;
       printf(“A2[%d][%d]=%d\n”,x,y,A2[x][y]);
     }

}
```

ผลการรันโปรแกรม

โปรแกรมจะกำหนดค่า 0 ถึง 9 ให้กับตัวแปรอาร์เรย์ 2 มิติ A2 ดังนี้

	[0]	[1]	[2]	[3]	[4]
A2[0][ ]	0	1	2	3	4
A2[1][ ]	5	6	7	8	9

```
E:\DevC\array2d.exe
A2[0][0]=0
A2[0][1]=1
A2[0][2]=2
A2[0][3]=3
A2[0][4]=4
A2[1][0]=5
A2[1][1]=6
A2[1][2]=7
A2[1][3]=8
A2[1][4]=9

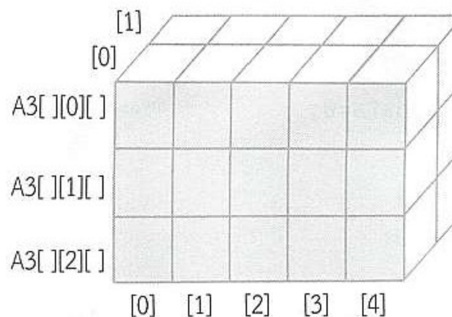
--------------------------------
Process exited after 0.02451 seconds with return value 0
Press any key to continue . . .
```

รูปที่ 4.4 ผลการรันโปรแกรมการกำหนดค่าให้ตัวแปรอาร์เรย์ 2 มิติ

ตัวอย่างการกำหนดตัวแปรอาร์เรย์แบบ 3 มิติ

char A3[2][3][5];

หมายถึง การประกาศตัวแปร A3 ให้เป็นตัวแปรอาร์เรย์ 3 มิติ แต่ละหน่วยเก็บข้อมูลได้ 8 บิต ซึ่งประกอบด้วยตัวแปรอาร์เรย์ A3[0][0][0] ถึง A3[1][2][4] จำนวน 30 หน่วย ดังรูปที่ 4.5



รูปที่ 4.5 ตัวแปรอาร์เรย์ขนาด 3 มิติ

ตัวอย่างที่ 4.3 โปรแกรมกำหนดค่า 0 ถึง 29 ให้กับตัวแปรอาร์เรย์ 3 มิติ

```
#inclide <stdio.h>
int main()
  { int A3[2][3][5],x,y,z,data=0;
     for(x=0;x<2;x++)
      for(y=0;y<3;y++)
       for(z=0;z<5;z++)
        { A3[x][y][z]=data;
          data=data+1;
          printf("A3[%d][%d][%d]=%d\n",x,y,z,A3[x][y][z]);
        }
  }
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบกำหนดค่า 0 ถึง 29 ให้กับตัวแปรอาร์เรย์ 3 มิติ A3 ดังนี้

```
A3[0][0][0]=0     A3[0][2][0]=10    A3[1][1][0]=20
A3[0][0][1]=1     A3[0][2][1]=11    A3[1][1][1]=21
A3[0][0][2]=2     A3[0][2][2]=12    A3[1][1][2]=22
A3[0][0][3]=3     A3[0][2][3]=13    A3[1][1][3]=23
A3[0][0][4]=4     A3[0][2][4]=14    A3[1][1][4]=24
A3[0][1][0]=5     A3[1][0][0]=15    A3[1][2][0]=25
A3[0][1][1]=6     A3[1][0][1]=16    A3[1][2][1]=26
A3[0][1][2]=7     A3[1][0][2]=17    A3[1][2][2]=27
A3[0][1][3]=8     A3[1][0][3]=18    A3[1][2][3]=28
A3[0][1][4]=9     A3[1][0][4]=19    A3[1][2][4]=29
```

4.2 การกำหนดค่าให้กับตัวแปรอาร์เรย์

การเขียนโปรแกรมบางครั้งจำเป็นต้องมีการกำหนดค่าเริ่มต้นให้กับตัวแปรอาร์เรย์ก่อน ซึ่งสามารถกำหนดค่าได้ดังนี้

ตัวอย่าง

```
char data[4] = {1,2,4,8};
```

หมายถึง กำหนดค่าให้กับตัวแปรอาร์เรย์ data[0] ถึง data[3] ให้มีค่าเท่ากับ 1, 2, 4, 8 ตามลำดับ

data[0]	data[1]	data[2]	data[3]
1	2	4	8

```
char text[4] = {'a','b','c','d'};
```

หมายถึง กำหนดค่าให้กับตัวแปรอาร์เรย์ text[0] ถึง text[3] ให้เก็บค่าตัวอักษร a, b, c, d ตามลำดับ

text[0]	text[1]	text[2]	text[3]
a	b	c	d

ตัวอย่างที่ 4.4 โปรแกรมแสดงตัวอักษรหรือชื่อโดยใช้ตัวแปรอาร์เรย์

```
1  #include <stdio.h>
2  int main()
3   { char name[20];
4     printf("What's your name : ");
5     scanf("%s",&name);
6     printf("Hello %s",name);
7   }
```

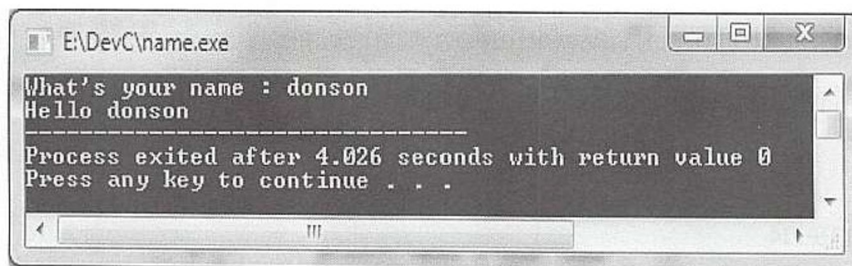
คำอธิบาย

บรรทัดที่ 4 พิมพ์ข้อความ What's your name :

บรรทัดที่ 5 รอรับข้อความ (%s, string) มาเก็บไว้ที่ตัวแปรอาร์เรย์ name เก็บสูงสุดได้ 19 ตัวอักษร

บรรทัดที่ 6 แสดงข้อความ Hello และตามด้วยข้อความในตัวแปรอาร์เรย์ name

ผลการรันโปรแกรม



รูปที่ 4.6 ผลการรันโปรแกรมแสดงตัวอักษรหรือชื่อโดยใช้ตัวแปรอาร์เรย์

ตัวอย่างที่ 4.5 โปรแกรมหาค่ามากที่สุดในตัวแปรอาร์เรย์ที่กำหนด

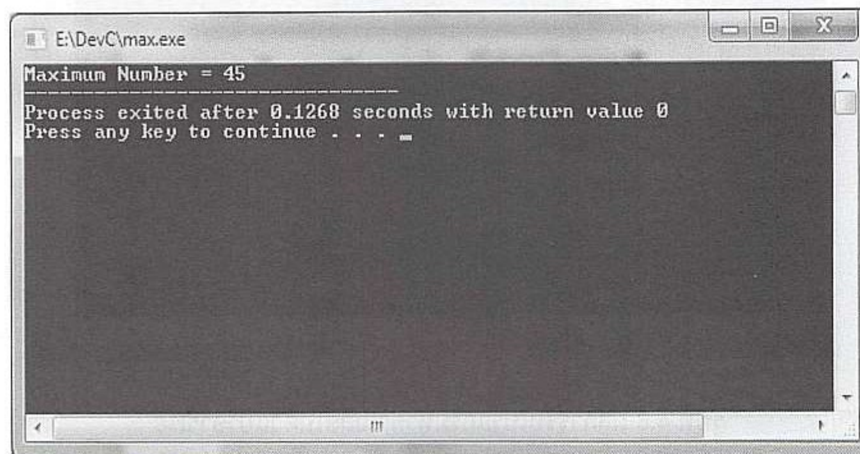
```
1   #include <stdio.h>
2   int main()
3   { int data[8]={10,7,18,45,33,23,2,42};
4     int max,i;
5     max=data[0];
6      for(i=1;i<8;i++)
7       {
8        if(data[i]>max)
9        max=data[i];
10      }
11     printf(“Maximum Number = %d”,max);
12  }
```

คำอธิบาย

บรรทัดที่ 3	ประกาศตัวแปรอาร์เรย์เป็นชนิดจำนวนเต็ม และกำหนดค่า data[0] ถึง data[7] เท่ากับ 10, 7, 18, 45, 33, 23, 2, 42
บรรทัดที่ 5	กำหนดค่าเริ่มต้นให้ตัวแปร max = data[0] = 10
บรรทัดที่ 6 ถึง 10	วนรอบ 7 ครั้ง เปรียบเทียบหาค่าสูงสุด โดยถ้า data[i] > max ให้นำค่ามาเก็บไว้ในตัวแปร max

ผลการรันโปรแกรม

โปรแกรมจะวนรอบ 7 ครั้งเพื่อค้นหาข้อมูลสูงสุดของอาร์เรย์ data[i] แล้วนำมาเก็บไว้ในตัวแปร max และแสดงข้อความ Maximum Number = 45 ดังรูป



รูปที่ 4.7 ผลการรันโปรแกรมหาค่ามากที่สุดในตัวแปรอาร์เรย์ที่กำหนด

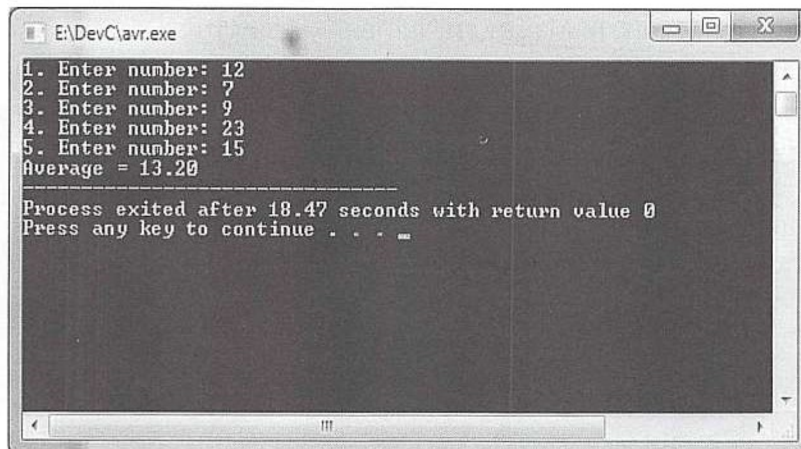
ตัวอย่างที่ 4.6 โปรแกรมหาค่าเฉลี่ยในตัวแปรอาร์เรย์

```
#include <stdio.h>
int main()
 { float sum=0,aver;
   int i,num[5];
   for(i=0;i<5;i++)
    {
     printf("%d. Enter number: ",i+1);
     scanf("%d",&num[i]);
     sum=sum+num[i];
    }
   aver=sum/5;
   printf("Average = %.2f",aver);
 }
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบ รับค่าตัวเลข 5 จำนวนมาเก็บไว้ในตัวแปรอาร์เรย์ num[i] แล้วบวกค่าทั้งหมดไว้ในตัวแปร sum จากนั้นคำนวณหาค่าเฉลี่ย (aver = sum/5) และแสดงค่าเฉลี่ยในรูปทศนิยม 2 ตำแหน่ง

```
Sum = 12 + 7 + 9 + 23 + 15 = 66
Average = 66/5 = 13.20
```



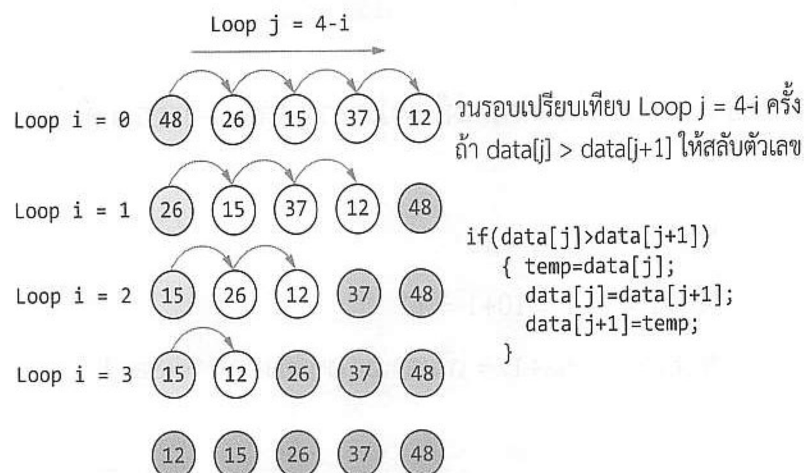
รูปที่ 4.8 ผลการรันโปรแกรมหาค่าเฉลี่ยในตัวแปรอาร์เรย์

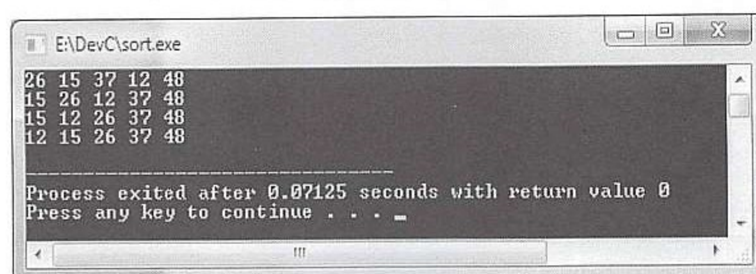
ตัวอย่างที่ 4.7 การเรียงข้อมูลในตัวแปรอาร์เรย์

```
#include <stdio.h>
int main()
 { int data[5]={48,26,15,37,12};
   int x,i,j,temp;
   for(i=0;i<4;i++)
    {
     for(j=0;j<4-i;j++)
      {if(data[j]>data[j+1])
         { temp=data[j];
           data[j]=data[j+1];
           data[j+1]=temp;
         }
      }
     for(x=0;x<5;x++)
     printf("%d ",data[x]);
     printf("\n");
    }
 }
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบเปรียบเทียบ ถ้า data[j] > data[j+1] ให้ทำการสลับข้อมูลโดยวนซ้ำ 4 รอบก็จะสามารถเรียงข้อมูลได้





รูปที่ 4.9 ผลการรันโปรแกรมการเรียงข้อมูลในตัวแปรอาร์เรย์

4.3 พอยเตอร์

พอยเตอร์ (Pointer) คือตัวชี้ตำแหน่งหรือแอดเดรสของข้อมูล ในการเก็บข้อมูลตัวแปรแบบพอยเตอร์จะเก็บตำแหน่งของข้อมูลแทนการเก็บค่าของข้อมูลจริง ๆ และการกำหนดชนิดของตัวแปรแบบพอยเตอร์จะใช้เครื่องหมาย "*" นำหน้าตัวแปรซึ่งมีรูปแบบดังนี้

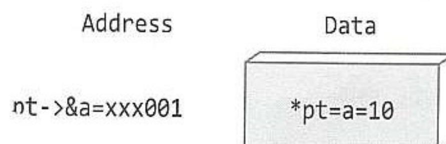
รูปแบบ

ชนิดตัวแปร *ชื่อตัวแปร;

ตัวอย่าง

```
int *pt, a = 10;
pt = &a;
```

หมายถึง ประกาศตัวแปร pt เป็นตัวแปรแบบพอยเตอร์ และ pt = &a; หมายถึงให้ pt เก็บแอดเดรสหรือตำแหน่งในหน่วยความจำของตัวแปร a ดังนั้น



pt = &a = แอดเดรสหรือตำแหน่งในหน่วยความจำของตัวแปร a ดังนั้น

*pt = *(&a) = a = 10

*pt+1 = a+1 = 10+1 = 11

*(pt+1) = *(&a+1) = เพิ่มตำแหน่งหรือแอดเดรส &a+1

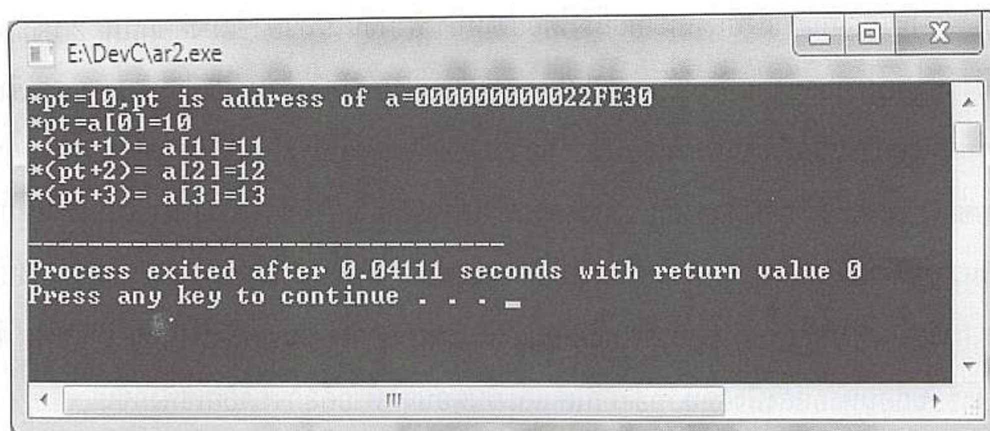
ตัวอย่างที่ 4.8 โปรแกรมการใช้งานตัวแปรพอยเตอร์กับอาร์เรย์

```
#include <stdio.h>
int main()
 {
   int *pt,a[4]={10,11,12,13};
   pt=a;
   printf("*pt=%d,pt is address of a=%p \n",*pt,pt);
   printf("*pt=a[0]=%d\n",*pt);
   printf("*(pt+1)= a[1]=%d\n",*(pt+1));
   printf("*(pt+2)= a[2]=%d\n",*(pt+2));
   printf("*(pt+3)= a[3]=%d\n",*(pt+3));
 }
```

ผลการรันโปรแกรม

โปรแกรมจะแสดงข้อมูลและตำแหน่งของตัวแปรพอยเตอร์ pt ดังนี้

Address of Memory	Data	
pt=&a[0]=22FE30	10	*pt=a[0]=10
pt+1=&a[1]=22FE31	11	*(pt+1)=a[1]=11
pt+2=&a[2]=22FE32	12	*(pt+2)=a[2]=12
pt+3=&a[3]=22FE33	13	*(pt+3)=a[3]=13



รูปที่ 4.10 แสดงการทำงานของตัวแปรพอยเตอร์กับอาร์เรย์

ตัวอย่างที่ 4.9 โปรแกรมค้นหาข้อมูลในอาร์เรย์โดยใช้พอยเตอร์

```
#include <stdio.h>
int main()
 {
   int data[8] ={10,7,18,45,33,23,2,42};
   int max,i,*pt;
   pt=data; // pt-->address data
   max=*pt;
   for(i=1;i<8;i++)
    {
     if(*(pt+i)>max)
     max=*(pt+i);
    }
   printf("Maximum Number = %d",max);
 }
```

ผลการรันโปรแกรม

โปรแกรมวนรอบ 7 ครั้งเพื่อค้นหาข้อมูลสูงสุดของอาร์เรย์ data[i] โดยใช้ตัวแปรพอยเตอร์เป็นตัวชี้ตำแหน่งของข้อมูลแล้วเก็บค่ามากสุดไว้ในตัวแปร max และแสดงข้อความดังนี้

Maximum Number = 45

4.4 การสร้างฟังก์ชัน

ฟังก์ชันคือคำสั่งที่ทำหน้าที่อย่างใดอย่างหนึ่ง ฟังก์ชันในภาษาซีแบ่งออกเป็น 2 ประเภท คือฟังก์ชันของภาษาซี เช่น ฟังก์ชัน printf() scanf() และฟังก์ชันที่สร้างหรือเขียนขึ้นเอง การเขียนฟังก์ชันหรือการเขียนโปรแกรมย่อยนับว่ามีความจำเป็นและมีประโยชน์ในการเขียนโปรแกรมที่ซับซ้อน เพราะจะช่วยให้การเขียนโปรแกรมทำได้ง่ายขึ้น ขนาดของโปรแกรมสั้นลงทำให้การประมวลผลทำได้เร็วขึ้น การเขียนโปรแกรมมีลักษณะเป็นลำดับขั้นทำให้สามารถแก้ไขโปรแกรมได้ง่าย การสร้างฟังก์ชันประกอบไปด้วยขั้นตอนดังนี้ เริ่มจากการประกาศชื่อฟังก์ชัน เขียนฟังก์ชันซึ่งจะเขียนก่อนหรือหลังฟังก์ชัน main ก็ได้ และขั้นตอนสุดท้ายคือการเรียกใช้ฟังก์ชัน

ตัวอย่างที่ 4.10 โปรแกรมการสร้างฟังก์ชันแสดงข้อความ

```
#include <stdio.h>
int sub1();
int main()
 {
  printf("************\n");
  sub1();
  printf("************\n");
 }
int sub1()
{
  printf("sub function\n");
}
```

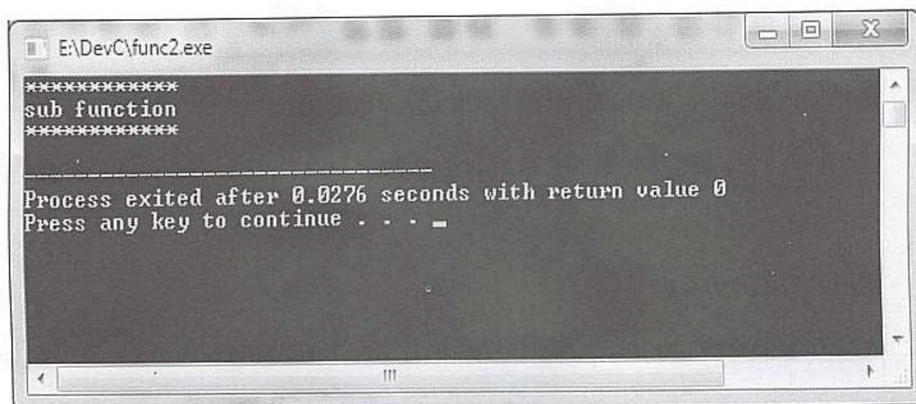
```
// ประกาศฟังก์ชัน sub1();

// เรียกใช้ฟังก์ชัน sub1();

// ฟังก์ชัน sub1() ไม่มีเซมิโคลอนปิดท้าย
```

ผลการรันโปรแกรม

โปรแกรมจะแสดงข้อความดังรูป



รูปที่ 4.11 ผลการรันโปรแกรมการสร้างฟังก์ชันแสดงข้อความ

4.5 การส่งค่าผ่านฟังก์ชัน

การส่งค่าผ่านฟังก์ชันคือการส่งค่าจากฟังก์ชันหนึ่งไปยังอีกฟังก์ชันหนึ่งซึ่งจะต้องกำหนดตัวแปรไว้ในการรับค่า การส่งค่าผ่านฟังก์ชันสามารถส่งไปค่าเดียวหรือหลายค่าก็ได้ และมีทั้งการส่งค่าไปและการส่งค่ากลับขึ้นอยู่กับลักษณะการใช้งาน การส่งค่าผ่านฟังก์ชันจะช่วยทำให้การเขียนโปรแกรมนั้นสั้นลงและง่ายต่อการแก้ไขโปรแกรม

ตัวอย่างที่ 4.11 โปรแกรมบวกเลขแบบส่งค่าผ่านฟังก์ชัน

```
1   #include <stdio.h>
2   int sum(int x);
3   int ans;
4   int main()
5    {ans=sum(5);
6     printf("Sum of Number=%d",ans);
7    }
8   int sum(int x)
9   {int y=0;
10    for(x=x;x>0;x--)
11      y=y+x;
12    return(y);
13  }
```

คำอธิบาย

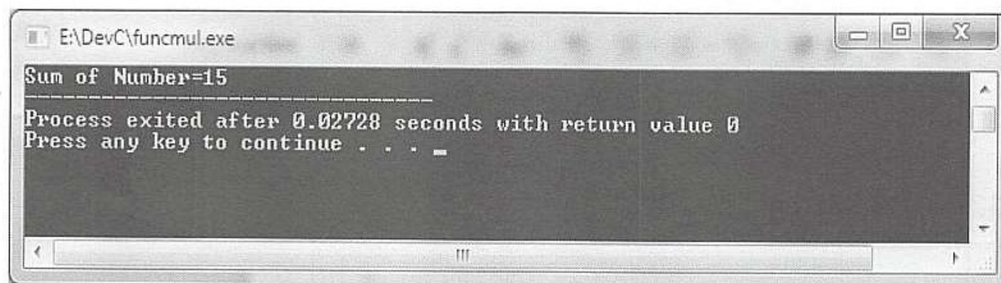
บรรทัดที่ 2 ประกาศฟังก์ชัน int sum(int x); ซึ่งมีตัวแปร x ใช้ในการรับและส่งค่า

บรรทัดที่ 5 เรียกใช้ฟังก์ชัน sum(5); และส่งค่า 5 ให้กับ x

บรรทัดที่ 8 ถึง 13 ฟังก์ชัน int sum(int x) ทำหน้าที่บวกเลข x = 5 ถึง 1 ตามค่าที่รับมา

ผลการรันโปรแกรม

โปรแกรมจะทำการบวกเลข 5 ถึง 1 ตามการส่งค่าของฟังก์ชัน sum(); โดยจะมีตัวแปร x รับค่าเข้ามาคำนวณ และมีการส่งค่า y กลับให้กับฟังก์ชัน sum(); และตัวแปร ans



รูปที่ 4.12 ผลการรันโปรแกรมบวกเลขแบบส่งค่าผ่านฟังก์ชัน

ตัวอย่างที่ 4.12 โปรแกรมการส่งค่าผ่านฟังก์ชันแบบหลายค่า

```
1   #include <stdio.h>
2   int mul(int a,int b);
3   int ans;
4   int main()
5     {
6      ans=mul(3,5);
7      printf("a*b = %d",ans);
8     }
9   int mul(int a,int b)
10  {
11    return(a*b);
12  }
```

คำอธิบาย

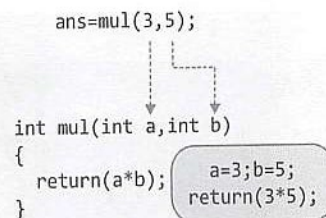
บรรทัดที่ 2 ประกาศฟังก์ชัน int mul(int a,int b); มีตัวแปร a และ b เป็นตัวแปรชนิดจำนวนเต็ม 16 บิตเพื่อใช้ในการรับและส่งค่า

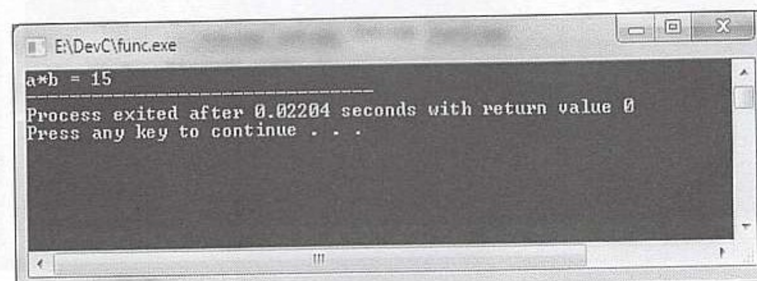
บรรทัดที่ 6 ans=mul(3,5); ส่งค่า 3 และ 5 ให้ฟังก์ชัน int mul(int a,int b)

บรรทัดที่ 9 ถึง 12 ฟังก์ชัน int mul(int a,int b) รับค่า 3 และ 5 มาเก็บไว้ที่ตัวแปร a และ b จากนั้นคูณ a กับ b แล้วจึงส่งค่ากลับ return(a*b);

ผลการรันโปรแกรม

โปรแกรมจะส่งค่า 3 และ 5 ให้ฟังก์ชัน mul(); เพื่อทำการคูณเลขแล้วส่งค่ากลับให้ฟังก์ชัน mul(); และตัวแปร ans ซึ่งมีค่าเท่ากับ 3 × 5 = 15





รูปที่ 4.13 ผลการรันโปรแกรมการส่งค่าผ่านฟังก์ชันแบบหลายค่า

4.6 ตัวแปรแบบโครงสร้างและยูเนียน

ตัวแปรแบบโครงสร้างเป็นตัวแปรที่สามารถเก็บข้อมูลหลาย ๆ ชนิดได้ เช่น ชื่อ ตัวเลข และตัวอักษร ส่วนใหญ่จะประยุกต์ใช้งานทางด้านฐานข้อมูล ส่วนตัวแปรแบบยูเนียนจะคล้ายกับตัวแปรแบบโครงสร้าง ต่างกันที่สมาชิกของตัวแปรแบบยูเนียนจะใช้หน่วยความจำร่วมกัน

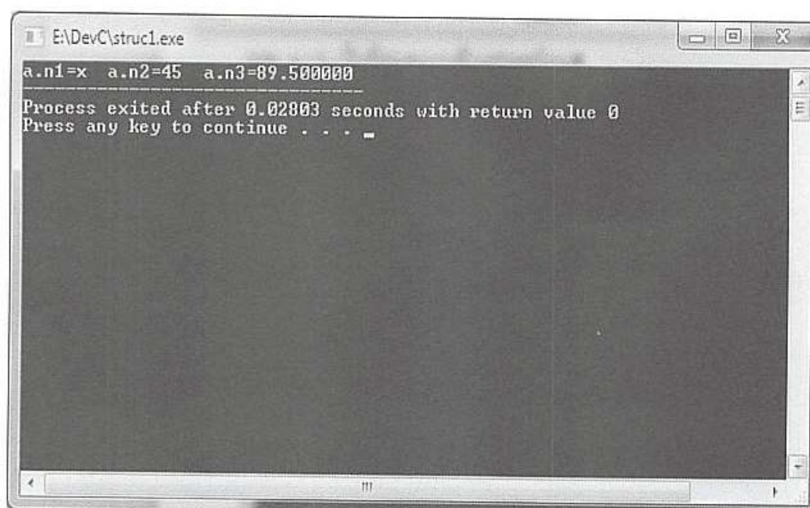
ตัวอย่างที่ 4.13 การประกาศตัวแปรแบบโครงสร้าง

```
#include <stdio.h>
struct
{ char n1;
  int n2;
  float n3;
}a;
int main()
 { a.n1='x';
   a.n2=45;
   a.n3=89.5;
   printf("a.n1=%c  a.n2=%d  a.n3=%f",a.n1,a.n2,a.n3);
 }
```

ผลการรันโปรแกรม

a เป็นตัวแปรแบบโครงสร้างโดยที่สมาชิก a.n1 เก็บข้อมูลแบบตัวอักษร a.n2 เก็บข้อมูลแบบตัวเลข และ a.n3 เก็บข้อมูลแบบตัวเลขทศนิยม

a.n1 = x   a.n2 = 45   a.n3 = 89.5



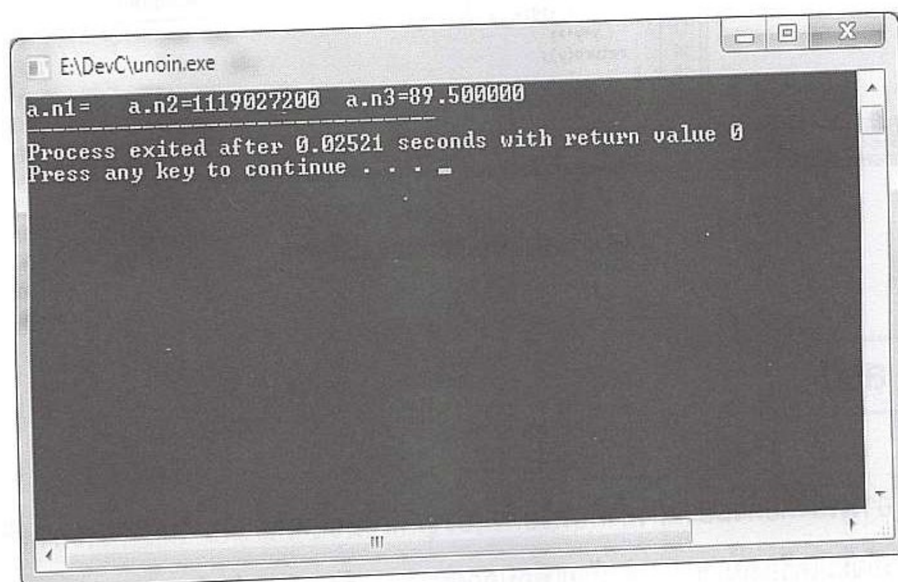
รูปที่ 4.14 ผลการรันโปรแกรมการประกาศตัวแปรแบบโครงสร้าง

ตัวอย่างที่ 4.14 การประกาศตัวแปรแบบยูเนียน

```
#include <stdio.h>
union
 { char n1;
   int n2;
   float n3;
 }a;
int main()
 { a.n1='x';
   a.n2=45;
   a.n3=89.5;
   printf("a.n1=%c  a.n2=%d  a.n3=%f",a.n1,a.n2,a.n3);
 }
```

ผลการรันโปรแกรม

a เป็นตัวแปรแบบยูเนียนโดยที่สมาชิก a.n1 เก็บข้อมูลแบบตัวอักษร a.n2 เก็บข้อมูลแบบตัวเลข และ a.n3 เก็บข้อมูลแบบตัวเลขทศนิยม แต่ทั้งหมดจะใช้หน่วยความจำร่วมกันจึงไม่สามารถเก็บข้อมูลพร้อมกันได้ (จะเก็บข้อมูลสุดท้ายคือ a.n3 = 89.5)



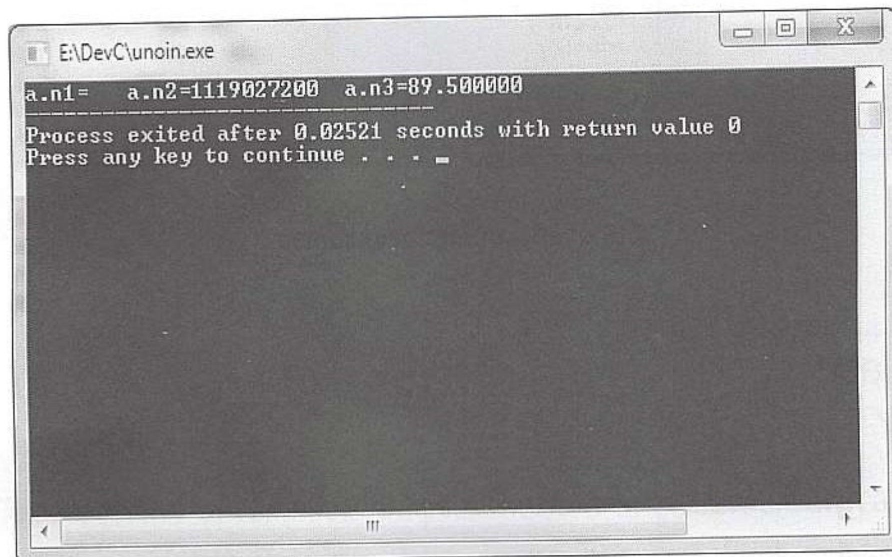
รูปที่ 4.15 ผลการรันโปรแกรมการประกาศตัวแปรแบบยูเนียน

ตัวอย่างที่ 4.14 การประกาศตัวแปรแบบยูเนียน

```
#include <stdio.h>
union
 { char n1;
   int n2;
   float n3;
 }a;
int main()
 { a.n1='x';
   a.n2=45;
   a.n3=89.5;
   printf("a.n1=%c  a.n2=%d  a.n3=%f",a.n1,a.n2,a.n3);
 }
```

ผลการรันโปรแกรม

a เป็นตัวแปรแบบยูเนียนโดยที่สมาชิก a.n1 เก็บข้อมูลแบบตัวอักษร a.n2 เก็บข้อมูลแบบตัวเลข และ a.n3 เก็บข้อมูลแบบตัวเลขทศนิยม แต่ทั้งหมดจะใช้หน่วยความจำร่วมกันจึงไม่สามารถเก็บข้อมูลพร้อมกันได้ (จะเก็บข้อมูลสุดท้ายคือ a.n3 = 89.5)

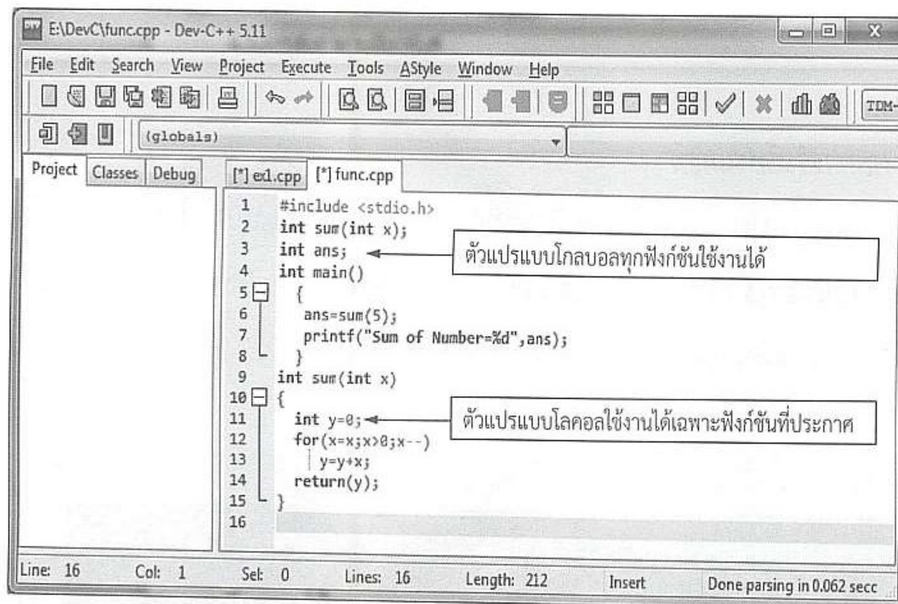


รูปที่ 4.15 ผลการรันโปรแกรมการประกาศตัวแปรแบบยูเนียน

4.7 ตัวแปรแบบโกลบอลและโลคอล

ตัวแปรแบบโกลบอล (Global) เป็นตัวแปรที่มีการประกาศก่อนฟังก์ชัน main() ซึ่งค่าเริ่มต้นของตัวแปรจะมีค่าเท่ากับศูนย์โดยอัตโนมัติ ตัวแปรแบบโกลบอลทุกฟังก์ชันสามารถใช้ร่วมกันได้โดยไม่ต้องประกาศตัวแปรในฟังก์ชันอีก

ตัวแปรแบบโลคอล (Local) เป็นตัวแปรที่มีการประกาศหลังฟังก์ชัน main หรือประกาศในฟังก์ชันที่สร้างขึ้น ซึ่งตัวแปรแบบโลคอลจะสามารถใช้ได้เฉพาะฟังก์ชันที่มีการประกาศตัวแปรเท่านั้น ฟังก์ชันที่ไม่ได้ประกาศจะไม่สามารถใช้งานได้



4.16 ตัวแปรแบบโกลบอลและโลคอล

4.8 สรุป

อาร์เรย์ คือตัวแปรแถวลำดับหรือตัวแปรชุดที่เป็นชนิดเดียวกัน ในการประกาศตัวแปรแบบอาร์เรย์จะมีเครื่องหมาย [ ] ต่อท้ายเพื่อบอกขนาดของตัวแปรอาร์เรย์

พอยเตอร์ คือตัวชี้ตำแหน่งหรือแอดเดรสของข้อมูล ในการเก็บข้อมูลตัวแปรแบบพอยเตอร์จะเก็บตำแหน่งของข้อมูลแทนการเก็บค่าของข้อมูลจริง ๆ และการกำหนดตัวแปรแบบพอยเตอร์จะใช้เครื่องหมาย “*” นำหน้าตัวแปร

ฟังก์ชันคือคำสั่งที่ทำหน้าที่อย่างใดอย่างหนึ่ง ฟังก์ชันในภาษาซีแบ่งออกเป็น 2 ประเภท คือฟังก์ชันของภาษาซี และฟังก์ชันที่สร้างหรือเขียนขึ้นเอง การใช้ฟังก์ชันจะช่วยทำให้การเขียนโปรแกรมง่ายขึ้นและลดขนาดของโปรแกรมให้น้อยลง

คำถามท้ายบทที่ 4

1. อาร์เรย์คืออะไร
2. พอยเตอร์คืออะไร
3. จงอธิบายการส่งค่าผ่านฟังก์ชัน
4. จงอธิบายหลักการเรียงข้อมูล
5. จงเขียนโปรแกรมค้นหาคะแนนต่ำสุดและสูงสุดของนักเรียนจำนวน 20 คน
6. จงเขียนโปรแกรมเรียงข้อมูลจากมากไปน้อยของเลข 10 จำนวน
7. จงเขียนโปรแกรมเรียงข้อมูลโดยใช้พอยเตอร์



พื้นฐาน Arduino



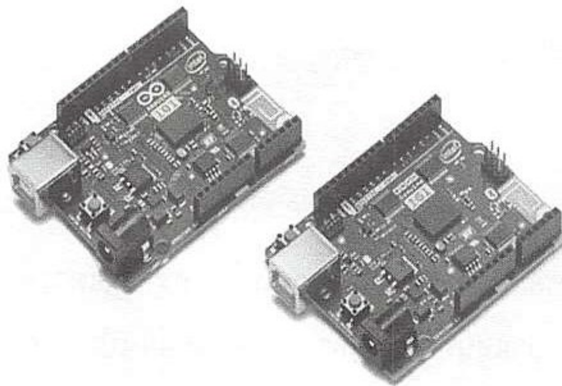
ในปัจจุบันวงจรอิเล็กทรอนิกส์ที่ประกอบไปด้วย ทรานซิสเตอร์ ตัวต้านทาน ตัวเก็บประจุ และไอซีดิจิทัลแบบเก่า ถูกแทนที่ด้วยไมโครคอนโทรลเลอร์ เนื่องจากไมโครคอนโทรลเลอร์มีประสิทธิภาพดีกว่า ทำงานในระบบซับซ้อนได้ดี ประมวลผลสัญญาณได้เร็ว มีการต่อวงจรที่ไม่ซับซ้อน สามารถเชื่อมต่อกับอุปกรณ์อื่น ๆ ได้ง่าย มีราคาถูก เขียนโปรแกรมสั่งงานได้หลายภาษา มีซอฟต์แวร์อำนวยความสะดวกเพื่อช่วยในการพัฒนาระบบได้เร็ว จึงทำให้ไมโครคอนโทรลเลอร์ได้รับความนิยมและเป็นส่วนประกอบที่สำคัญในวงจรอิเล็กทรอนิกส์และระบบควบคุมอัตโนมัติ

5.1 ไมโครคอนโทรลเลอร์

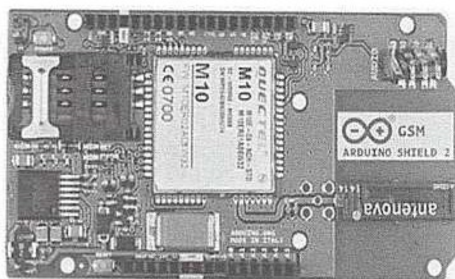
ไมโครคอนโทรลเลอร์ (Microcontroller) คือตัวควบคุมขนาดเล็ก ใช้เทคโนโลยีการผลิตระดับไมโคร ซึ่งมีหลากหลายตระกูล เช่น MCS-51 AVR PIC และ 68HC ไมโครคอนโทรลเลอร์เป็นอุปกรณ์อิเล็กทรอนิกส์ชนิดหนึ่งซึ่งมีลักษณะโครงสร้างเป็นไอซี (IC : Integrated Circuit) หรือชิป (Chip) ซึ่งทำหน้าที่ประมวลผลตามโปรแกรมหรือชุดคำสั่ง โครงสร้างภายในจะเป็นวงจรรวมขนาดใหญ่ประกอบไปด้วย หน่วยคำนวณทางคณิตศาสตร์และลอจิก บัสข้อมูล บัสควบคุม บัสที่อยู่ พอร์ตขนาน พอร์ตอนุกรม รีจิสเตอร์ หน่วยความจำ วงจรนับ วงจรจับเวลา และวงจรอื่น ๆ รวมกันอยู่ภายในตัวไอซี ไมโครคอนโทรลเลอร์ถูกออกแบบมาเพื่อใช้ในงานควบคุม สามารถติดต่อกับอุปกรณ์อินพุตและเอาต์พุตได้สะดวก ใช้งานง่าย สามารถทำงานได้โดยใช้ชิปเดียว มีคำสั่งที่สนับสนุนการเขียนโปรแกรมควบคุมและสามารถเข้าถึงข้อมูลระดับบิตได้

5.2 Arduino คืออะไร ?

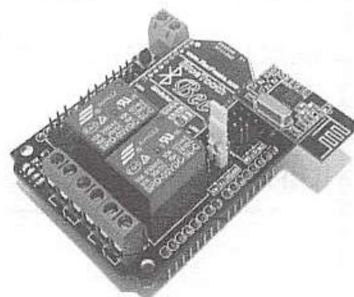
Arduino (อาดูอิโน่ เป็นภาษาอิตาเลียน) คือไมโครคอนโทรลเลอร์ขนาด 8 บิตในตระกูล AVR ที่ถูกออกแบบมาเพื่อให้ใช้งานง่าย สามารถเชื่อมต่อกับคอมพิวเตอร์ด้วยพอร์ต USB มีบอร์ดอุปกรณ์เชื่อมต่อหรือ Arduino Shield มากมายทำให้สะดวกในการพัฒนางาน การออกแบบและพัฒนาได้เปิดเผยข้อมูลทั้งด้านฮาร์ดแวร์และซอฟต์แวร์ (Open Source) จึงมีผู้ใช้งานจำนวนมาก Arduino ถูกออกแบบมาให้ใช้งานได้ง่ายทั้งฮาร์ดแวร์และการเขียนโปรแกรม จึงเหมาะสำหรับทุกคนที่สนใจเรียนรู้และต้องการประยุกต์ใช้งานไมโครคอนโทรลเลอร์



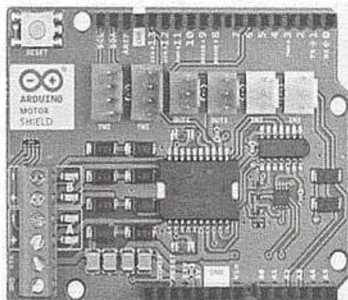
รูปที่ 5.1 บอร์ด Arduino



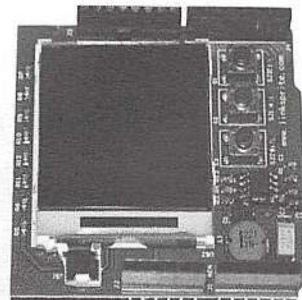
GSM Shield



Relay Shield with Wireless Interface



Motor Shield



Color Image LCD Shield

รูปที่ 5.2 Arduino Shield แบบต่าง ๆ

ด้วยความสะดวกและง่ายของ Arduino ในการเขียนโปรแกรมและการเชื่อมต่อกับอุปกรณ์อินพุต-เอาต์พุต ทั้งยังมีแอนะล็อกอินพุตราคาถูก ซอฟต์แวร์ฟรี จึงส่งผลให้บอร์ด Arduino เป็นที่นิยมอย่างแพร่หลายและมีบอร์ดหลากหลายรุ่น ตัวอย่างเช่น

1) Arduino Uno R3

เป็นบอร์ด Arduino ที่เป็นไมโครคอนโทรลเลอร์ขนาด 8 บิต ได้รับความนิยมอย่างมาก เนื่องจากใช้งานง่าย ราคาไม่แพง สามารถถอดเปลี่ยนไอซีไมโครคอนโทรลเลอร์ได้ ไลบรารีที่พัฒนาขึ้นส่วนใหญ่รองรับกับ Arduino Uno และยังมีอุปกรณ์เชื่อมต่อที่รองรับมากกว่ารุ่นอื่น ๆ



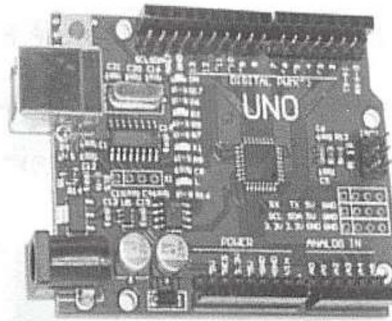
รูปที่ 5.3 บอร์ด Arduino UNO R3

คุณลักษณะของบอร์ด

ไมโครคอนโทรลเลอร์	ATmega328P
แรงดันไฟฟ้า	5 V
แรงดันไฟฟ้าอินพุต (แนะนำ)	7–12 V
แรงดันไฟฟ้าอินพุต (จำกัด)	6–20 V
ดิจิทัลอินพุตและเอาต์พุต	14 บิต (6 เอาต์พุต PWM)
แอนะล็อกอินพุต	6 บิต
แรงดันและกระแสไฟที่จ่ายได้ในแต่ละบิต	5 V, 40 mA
แรงดันและกระแสไฟที่จ่ายได้ในแต่ละบิต	3.3 V, 50 mA
หน่วยความจำแบบแฟลช (Flash)	32 KB (500 B Boot Loader)
หน่วยความจำแรม (RAM)	2 KB
หน่วยความจำรอม (ROM)	1 KB
ความถี่	16 MHz
ขนาด	68.6 x 53.4 mm
น้ำหนัก	25 กรัม

2) Arduino Uno SMD

เป็นบอร์ดที่มีคุณสมบัติการทำงานและขาสัญญาณเหมือนกับบอร์ด Arduino UNO R3 ทุกประการ แต่จะแตกต่างกันที่ตัวไอซีของไมโครคอนโทรลเลอร์ที่ติดอยู่ในบอร์ด ซึ่งจะไม่สามารถถอดเปลี่ยนได้ในกรณีที่ไอซีไมโครคอนโทรลเลอร์เสีย



รูปที่ 5.4 บอร์ด Arduino Uno SMD

คุณลักษณะของบอร์ด

ไมโครคอนโทรลเลอร์	ATmega328P
แรงดันไฟฟ้า	5 V
แรงดันไฟฟ้าอินพุต (แนะนำ)	7–12 V
แรงดันไฟฟ้าอินพุต (จำกัด)	6–20 V
ดิจิทัลอินพุตและเอาต์พุต	14 บิต (6 เอาต์พุต PWM)
แอนะล็อกอินพุต	6 บิต
แรงดันและกระแสไฟที่จ่ายได้ในแต่ละบิต	5 V, 40 mA
แรงดันและกระแสไฟที่จ่ายได้ในแต่ละบิต	3.3 V, 50 mA
หน่วยความจำแบบแฟลช	32 KB (500 B Boot Loader)
หน่วยความจำแรม	2 KB
หน่วยความจำรอม	1 KB
ความถี่	16 MHz
ขนาด	68.6 x 53.4 mm

3) Arduino Mega 2560

เป็นบอร์ดที่ออกแบบมาสำหรับงานที่ต้องใช้อุปกรณ์อินพุตและเอาต์พุตจำนวนมาก โดยมีขาสัญญาณ 54 พอร์ตดิจิทัลอินพุต-เอาต์พุต 15 เอาต์พุตพอร์ต PWM และ 16 พอร์ตแอนะล็อกอินพุต เหมาะสำหรับงานที่มีอุปกรณ์อินพุตและเอาต์พุตจำนวนมากหรืองานที่ใช้เซอร์โวมอเตอร์หลาย ๆ ตัว นอกจากนี้บอร์ด Arduino Mega 2560 ยังมีความหน่วยความจำแบบแฟลชมากถึง 256 KB ทำให้สามารถเขียนโปรแกรมได้มากกว่า

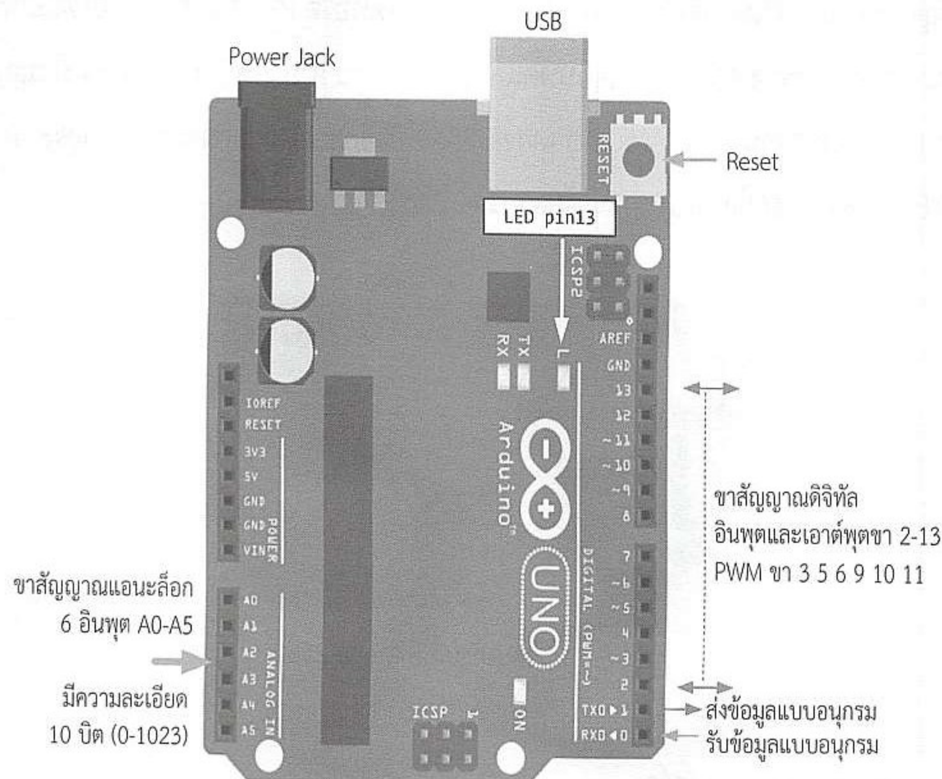


รูปที่ 5.5 บอร์ด Arduino Mega 2560

คุณลักษณะของบอร์ด

ไมโครคอนโทรลเลอร์	ATmega2560
แรงดันไฟฟ้า	5 V
แรงดันไฟฟ้าอินพุต (แนะนำ)	7–12 V
แรงดันไฟฟ้าอินพุต (จำกัด)	6–20 V
ดิจิทัลอินพุตและเอาต์พุต	54 บิต (15 เอาต์พุต PWM)
แอนะล็อกอินพุต	16 บิต
แรงดันและกระแสไฟที่จ่ายได้ในแต่ละบิต	5 V, 20 mA
แรงดันและกระแสไฟที่จ่ายได้ในแต่ละบิต	3.3 V, 50 mA
หน่วยความจำแบบแฟลช	256 KB (8 KB Boot Loader)
หน่วยความจำแรม	8 KB
หน่วยความจำรอม	4 KB
ความถี่	16 MHz
ขนาด	101.52 x 53.3 mm
น้ำหนัก	37 กรัม

5.3 ขาสัญญาณของบอร์ด Arduino UNO



รูปที่ 5.6 ขาสัญญาณของบอร์ด Arduino UNO

หน้าที่ของขาสัญญาณของบอร์ด Arduino

IOREF คือ ขาสัญญาณแรงดันไฟอ้างอิงเมื่อเชื่อมต่ออุปกรณ์อินพุตและเอาต์พุต

Reset คือ ขารีเซ็ตการทำงานของไมโครคอนโทรลเลอร์

3V3 คือ ขาสัญญาณแรงดันไฟ 3.3 V

5V คือ ขาสัญญาณแรงดันไฟ 5 V

GND คือ ขากราวด์

VIN คือ ขาแรงดันไฟอินพุตที่ป้อนให้บอร์ด

A0-A5 คือ ขาสัญญาณแอนะล็อกอินพุต มี 6 ขา คือ A0 A1 A2 A3 A4 และ A5 มีความละเอียด 10 บิต

RXD คือ ขาสัญญาณรับข้อมูลของการสื่อสารพอร์ตอนุกรม

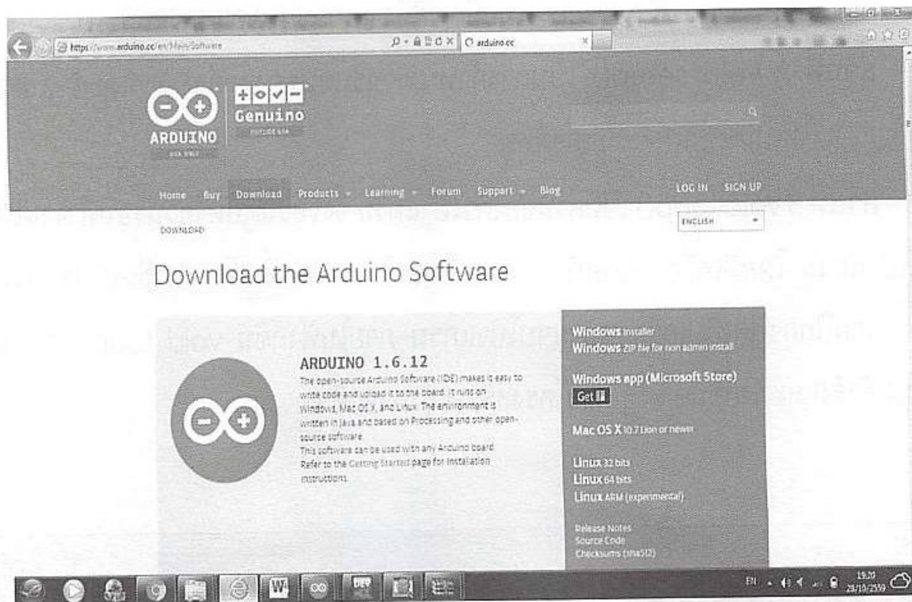
TXD คือ ขาสัญญาณส่งข้อมูลของการสื่อสารพอร์ตอนุกรม

2-13 คือ ขาสัญญาณดิจิทัลอินพุตและเอาต์พุต และมีเอาต์พุต PWM 6 ขา คือ ขา 3 5 6 9 10 และ 11

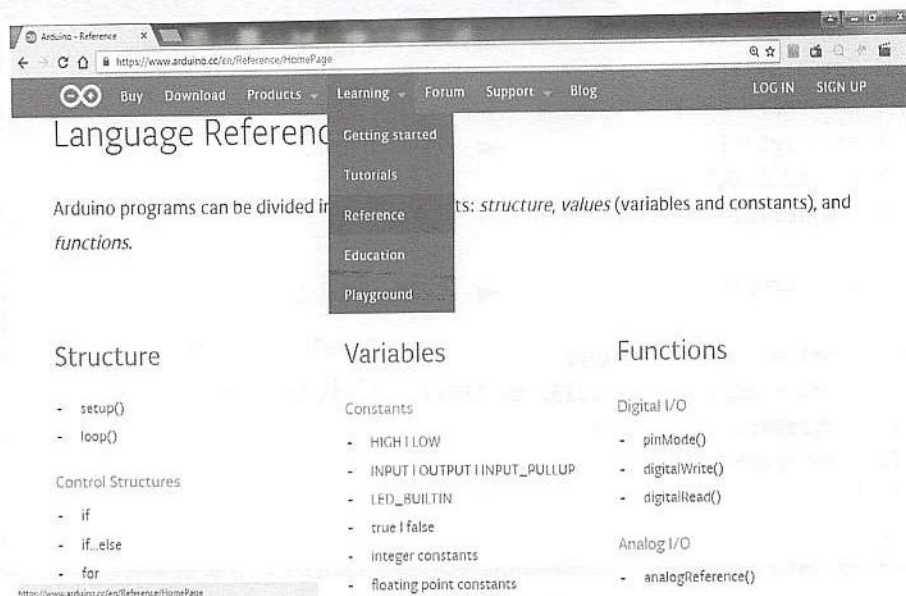
AREF คือ ขาสัญญาณแรงดันไฟอ้างอิงสำหรับแอนะล็อกอินพุต

5.4 โปรแกรม Arduino

หนังสือเล่มนี้ใช้โปรแกรม Arduino 1.6.12 ซึ่งเป็นซอฟต์แวร์ฟรี สามารถดาวน์โหลดโปรแกรมได้ที่ www.arduino.cc โดยต้องติดตั้งไดรเวอร์บอร์ด Arduino ด้วย เมื่อดาวน์โหลดเสร็จก็ติดตั้งได้เลย และสามารถเรียนรู้คำสั่งต่าง ๆ เพิ่มได้ที่เมนู Reference



รูปที่ 5.7 เว็บไซต์ arduino.cc



รูปที่ 5.8 เมนู Reference สำหรับการอ้างอิงคำสั่งต่าง ๆ

โปรแกรม Arduino แบ่งได้เป็น 3 ส่วนคือ 1) ไฟล์ส่วนหัวโปรแกรม ซึ่งจะเขียนก็ต่อเมื่อมีการเรียกใช้ไลบรารี 2) void setup() เป็นส่วนของการกำหนดค่า และ 3) void loop() เป็นส่วนของตัวโปรแกรมซึ่งจะวนรอบตลอดกาลเมื่อรันโปรแกรม แต่ละส่วนมีรายละเอียดดังนี้

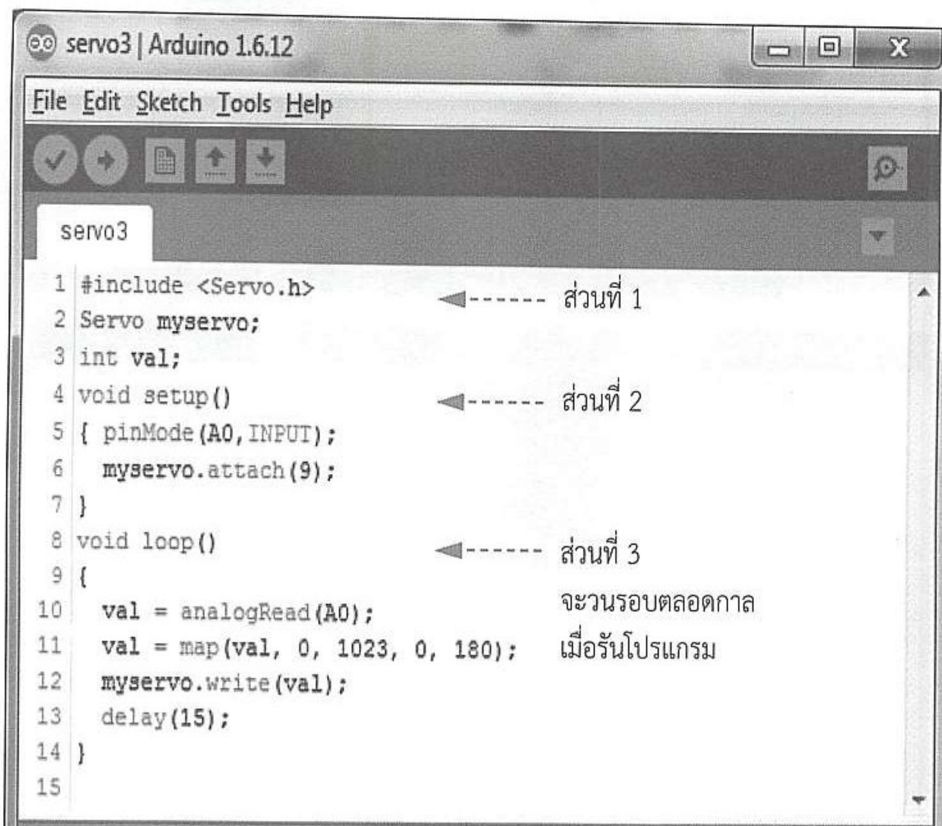
ส่วนที่ 1 ไฟล์ส่วนหัวโปรแกรม ซึ่งจะมีหรือไม่มีก็ได้ เป็นไฟล์ที่มีส่วนขยายเป็น *.h เพื่อเรียกใช้งานไลบรารีต่าง ๆ ของ Arduino ตัวอย่างเช่น

#include <Servo.h> เมื่อมีการใช้งานเกี่ยวกับเซอร์โวมอเตอร์

#include <LiquidCrystal.h> เมื่อมีการใช้งานเกี่ยวกับจอแสดงผล LCD

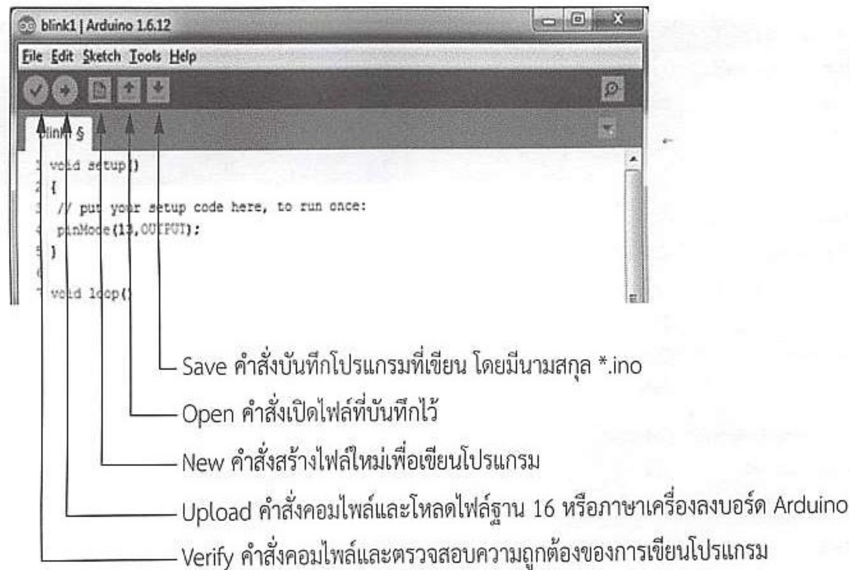
ส่วนที่ 2 void setup() เป็นการกำหนดค่าเริ่มต้นของค่าต่าง ๆ และกำหนดสถานะของขาสัญญาณให้เป็นขาอินพุตหรือเอาต์พุต

ส่วนที่ 3 void loop() เป็นส่วนของตัวโปรแกรม ซึ่งจะวนลูปหรือวนรอบทำงานไปตลอดเมื่อรันโปรแกรม โดยมีเครื่องหมายปีกกาเปิดเป็นเครื่องหมายเริ่มต้นการเขียนโปรแกรม และมีเครื่องหมายปีกกาปิดเป็นเครื่องหมายจบโปรแกรม ภายในฟังก์ชัน void loop() จะประกอบไปด้วยชุดคำสั่งและฟังก์ชันต่าง ๆ ตามโครงสร้างของภาษาซี



รูปที่ 5.9 ส่วนประกอบของโปรแกรม Arduino

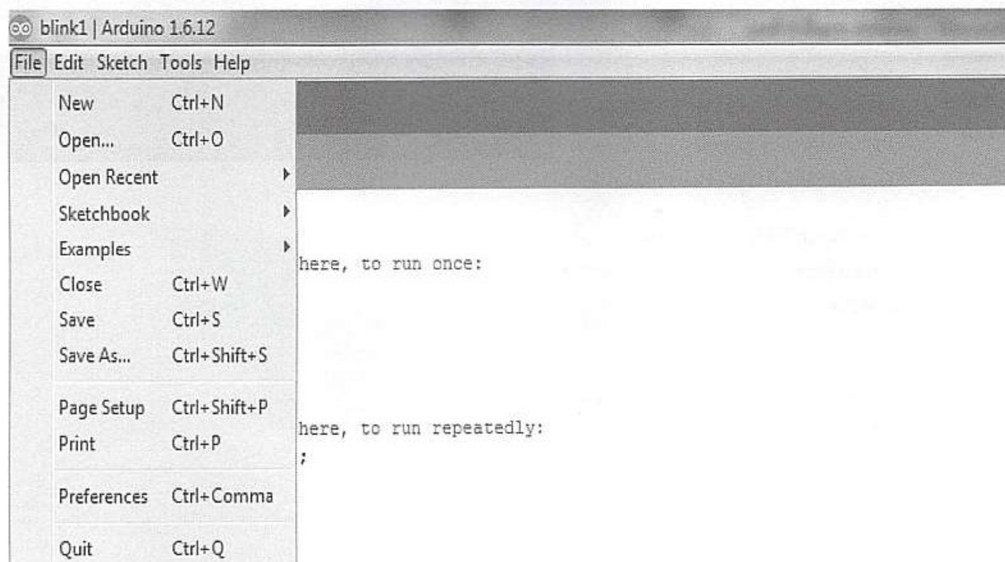
5.5 เมนูคำสั่งของโปรแกรม Arduino



รูปที่ 5.10 เมนูคำสั่งหลักของ Arduino

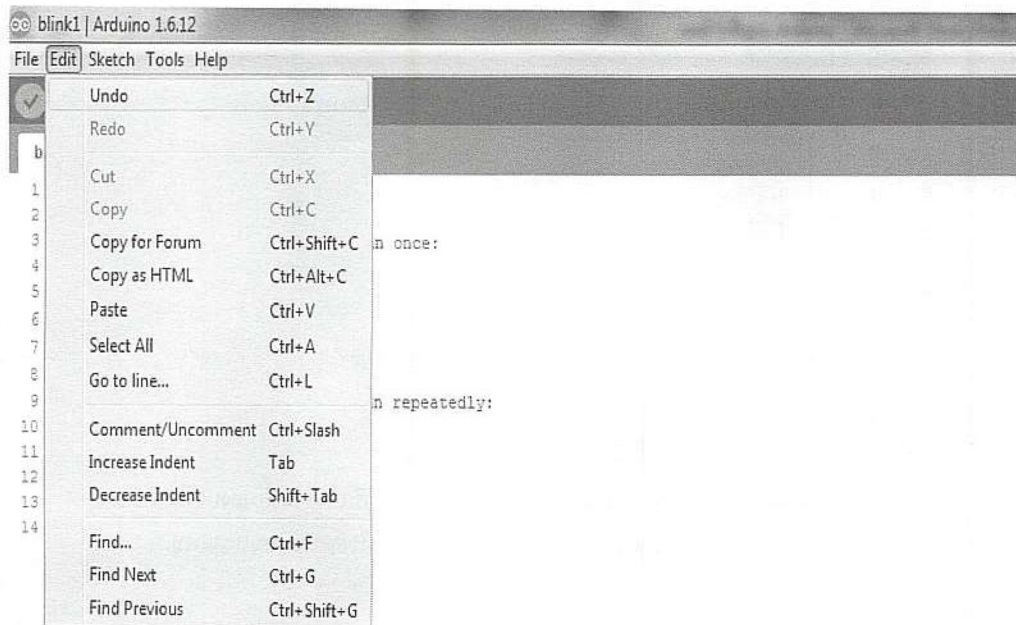
เมนูคำสั่งของ Arduino ประกอบด้วยเมนูคำสั่ง File, Edit, Sketch, Tools และ Help มีรายละเอียดดังนี้

1) **เมนู File** เป็นเมนูคำสั่งในการจัดการต่าง ๆ เกี่ยวกับไฟล์ เช่น การเปิดไฟล์ บันทึกไฟล์ การพิมพ์งาน และออกจากโปรแกรม



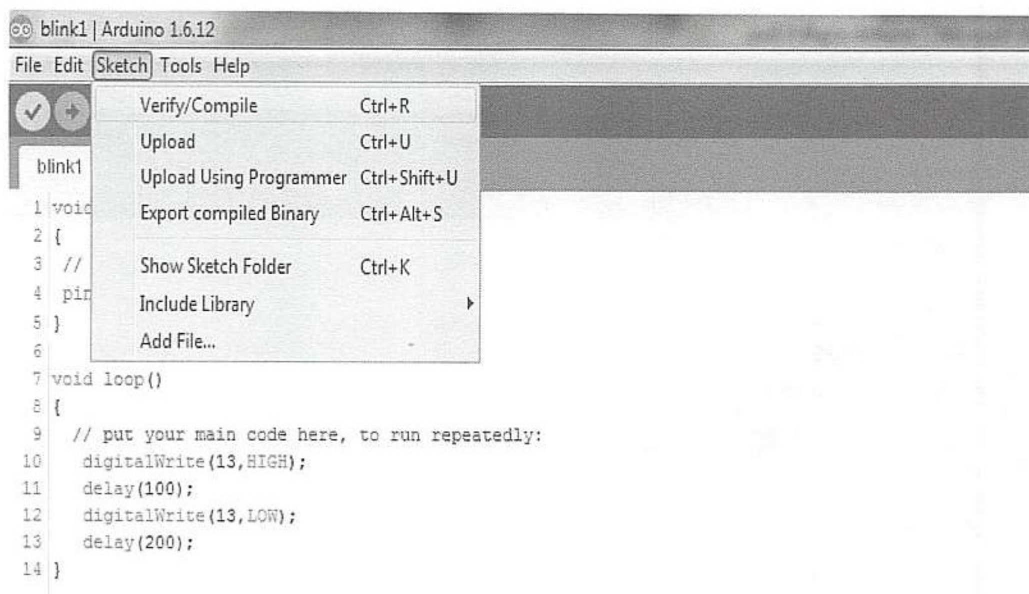
รูปที่ 5.11 เมนูคำสั่ง File

2) เมนู Edit เป็นเมนูคำสั่งเกี่ยวกับการแก้ไข เช่น การคัดลอก การวาง การยกเลิก และการค้นหา



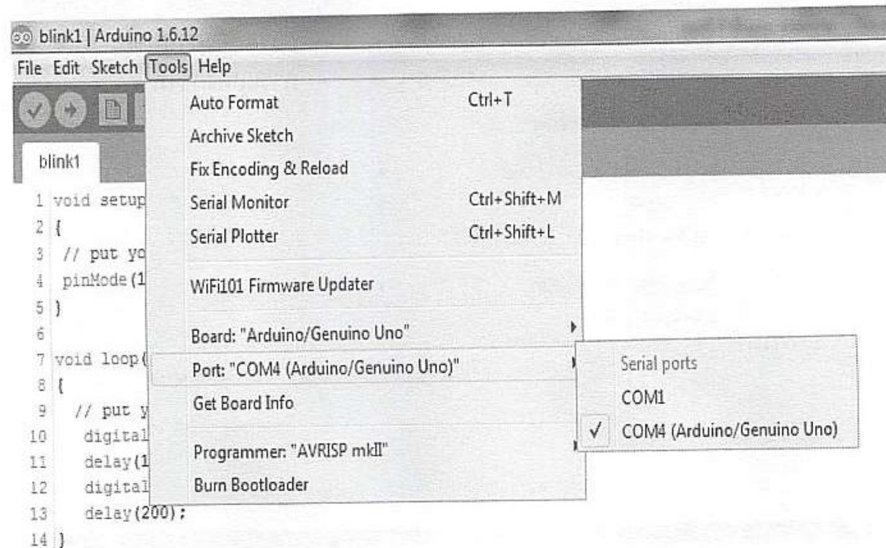
รูปที่ 5.12 เมนูคำสั่ง Edit

3) เมนู Sketch เป็นเมนูคำสั่งที่เกี่ยวข้องกับการคอมไพล์ การตรวจสอบความถูกต้องของการเขียนโปรแกรม และการโหลดโปรแกรมลงบอร์ด Arduino

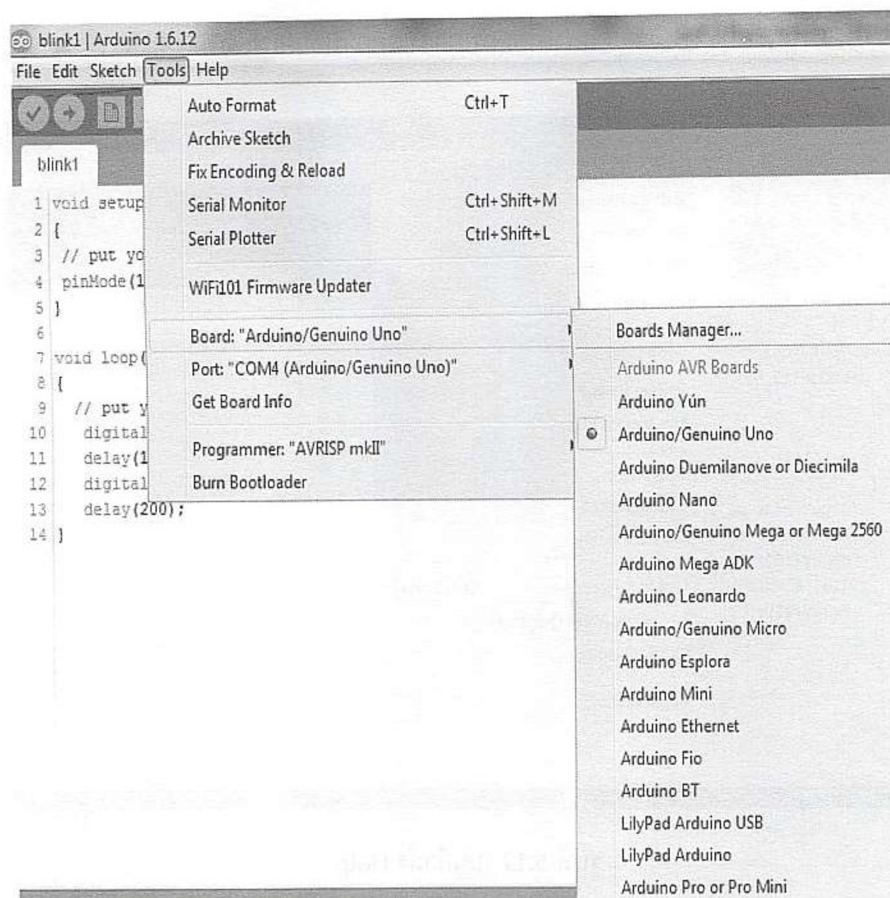


รูปที่ 5.13 เมนูคำสั่ง Sketch

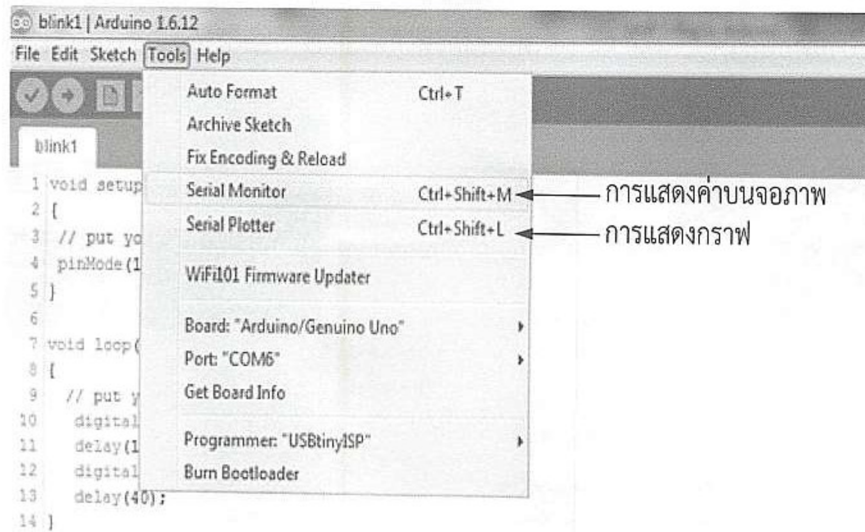
4) เมนู Tools เป็นเมนูคำสั่งในการใช้เครื่องมือต่าง ๆ สิ่งที่ต้องกำหนดคือพอร์ตอนุกรมสำหรับเครื่องนี้คือพอร์ต COM4 และกำหนดบอร์ด Arduino ที่ใช้คือ Arduino/Genuino Uno ซึ่งต้องกำหนดให้ตรงกับบอร์ดที่ใช้



รูปที่ 5.14 การกำหนดพอร์ต COM

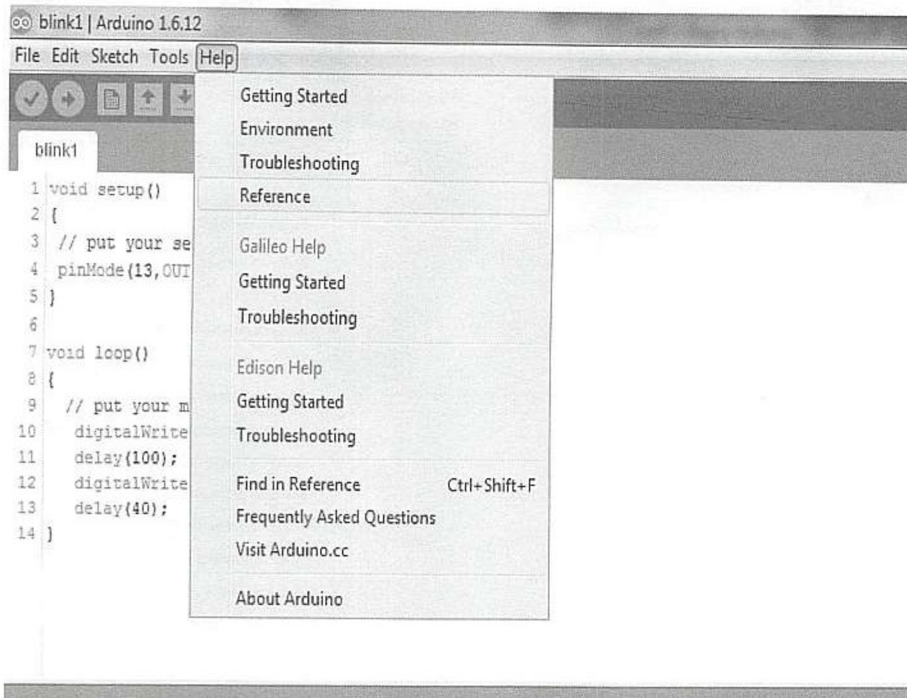


รูปที่ 5.15 การกำหนดบอร์ด Arduino/Genuino Uno



รูปที่ 5.16 การใช้คำสั่ง Serial Monitor และ Serial Plotter

5) เมนู Help เป็นเมนูคำสั่งเกี่ยวกับการขอความช่วยเหลือ

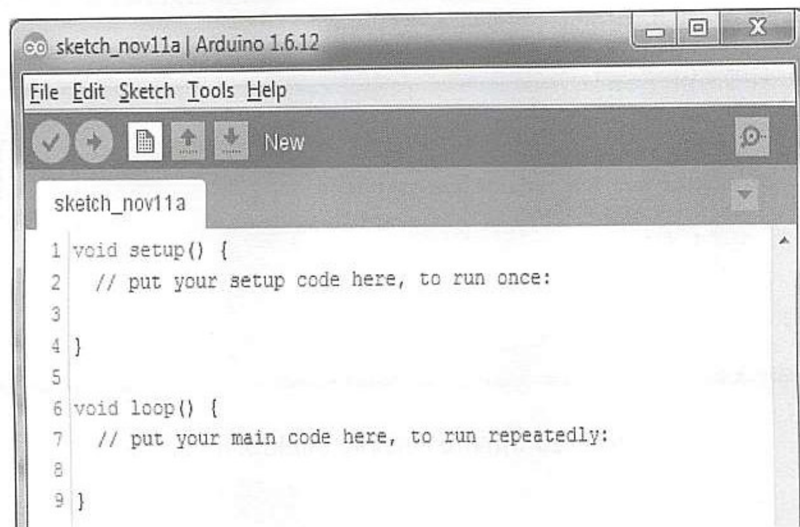


รูปที่ 5.17 เมนูคำสั่ง Help

5.6 การใช้งานโปรแกรม

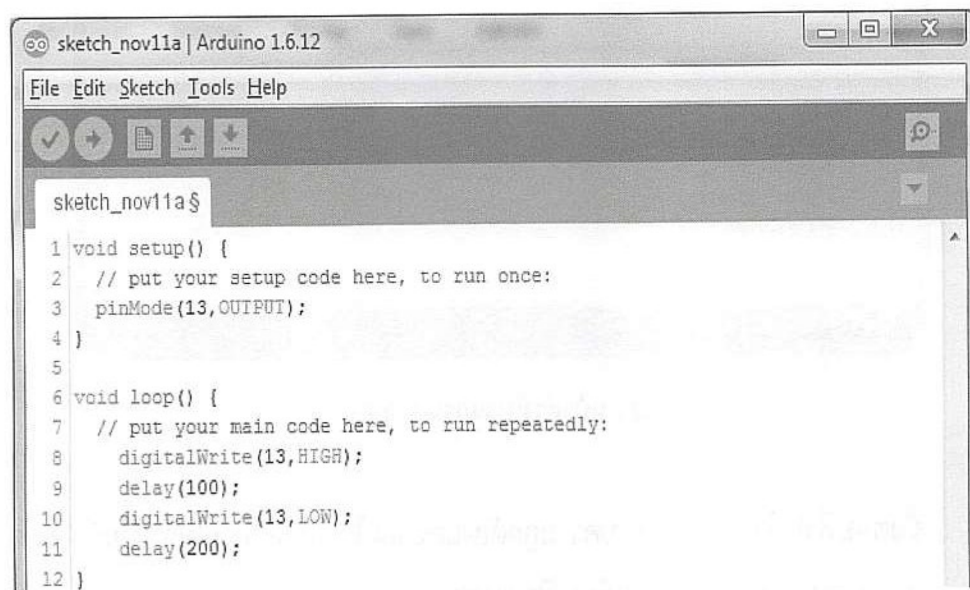
การใช้งานโปรแกรม Arduino เริ่มจากการสร้างไฟล์ใหม่ เขียนโปรแกรมตามโครงสร้างของภาษาซี คอมไพล์และโหลดโปรแกรมลงบนบอร์ด Arduino ซึ่งมีขั้นตอนดังนี้

ขั้นตอนที่ 1 เข้าโปรแกรม Arduino แล้วกด New จะได้หน้าต่างไฟล์ใหม่ดังนี้



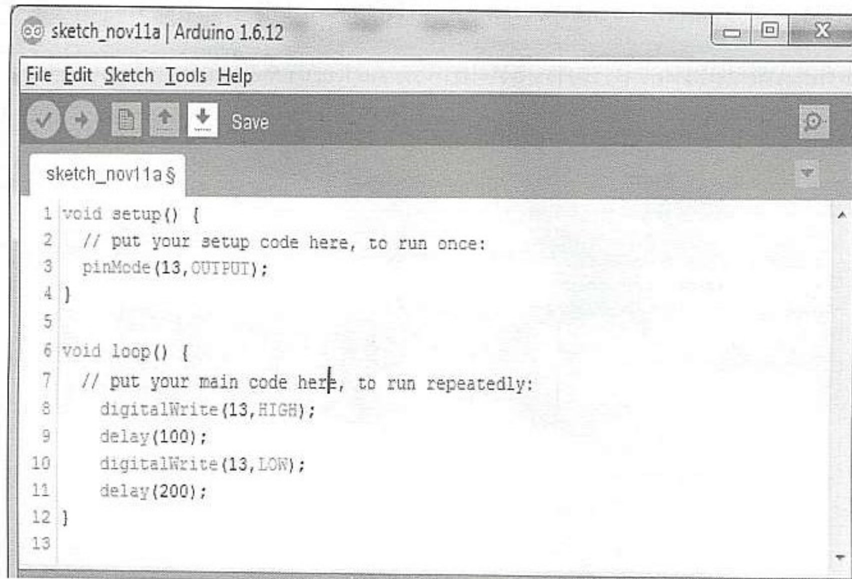
รูปที่ 5.18 หน้าต่างโปรแกรม Arduino

ขั้นตอนที่ 2 พิมพ์โปรแกรมไฟกะพริบ

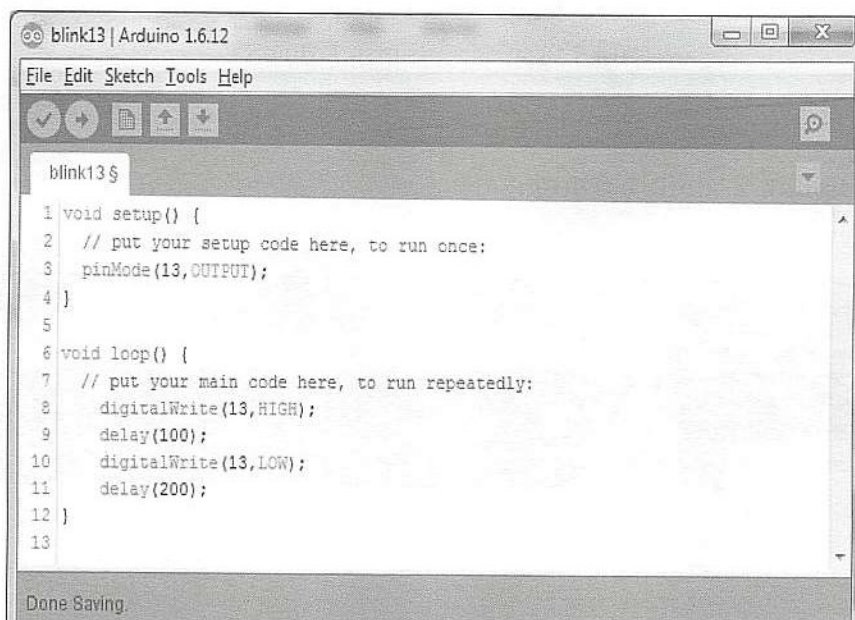


รูปที่ 5.19 เริ่มเขียนโปรแกรม

ขั้นตอนที่ 3 ทำการบันทึกโปรแกรมในชื่อ blink13.ino เมื่อบันทึกเสร็จโปรแกรมจะเปลี่ยนชื่อเป็น blink13

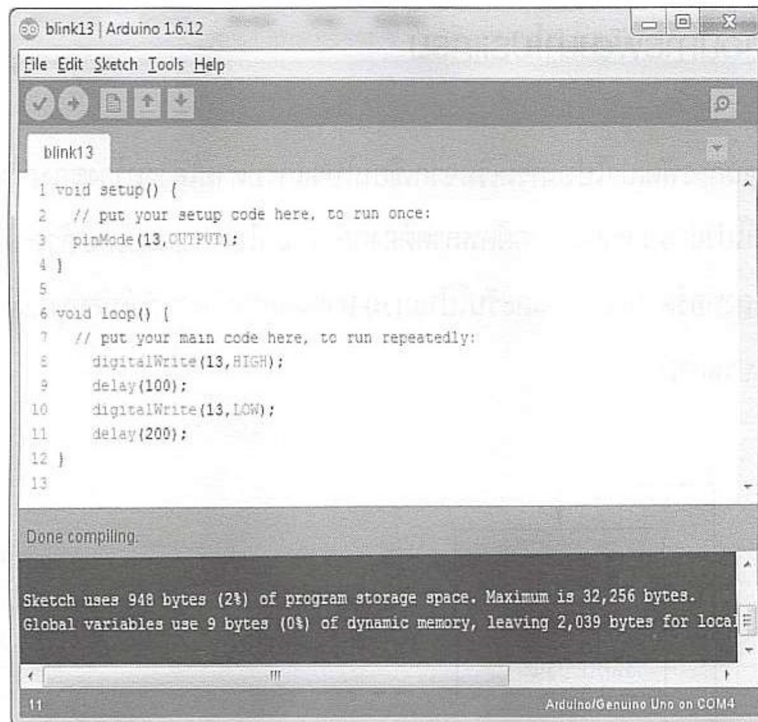


รูปที่ 5.20 บันทึกโปรแกรมชื่อ blink13.ino



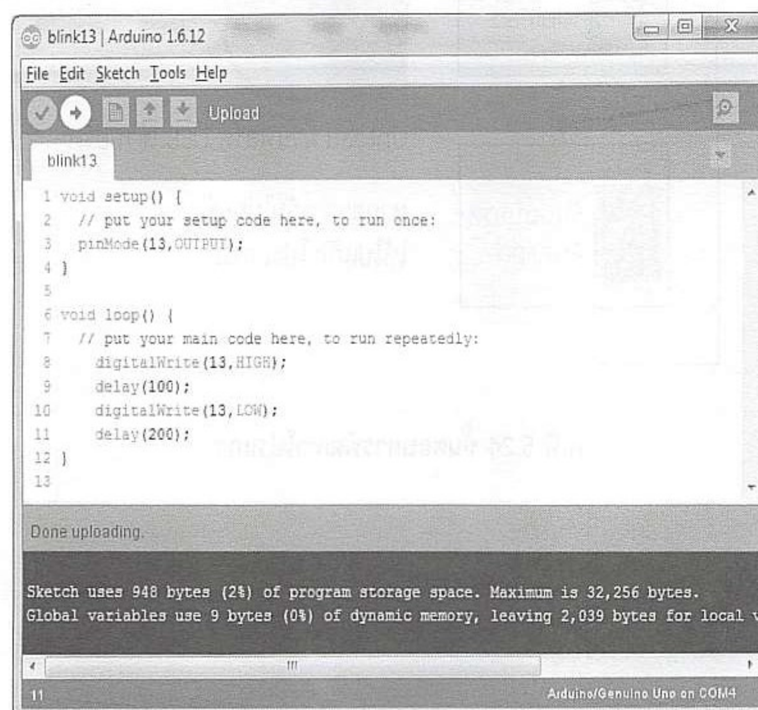
รูปที่ 5.21 หน้าต่างโปรแกรม blink13.ino

ขั้นตอนที่ 4 ทำการตรวจสอบความถูกต้องและคอมไพล์โปรแกรม ไม่จำเป็นต้องมีบอร์ด Arduino หากเกิดข้อผิดพลาดต้องกลับไปแก้ไขที่โปรแกรม



รูปที่ 5.22 ผลการคอมไพล์โปรแกรม

ขั้นตอนที่ 5 คอมไพล์โปรแกรมและโหลดโปรแกรมลงบอร์ด Arduino (สามารถข้ามขั้นตอนที่ 4 ได้) หากเกิดข้อผิดพลาดต้องกลับไปแก้ไขที่โปรแกรมหรือไปแก้ไขที่พอร์ตอนุกรม สังเกตไฟที่บอร์ดจะกะพริบแสดงว่ากำลังโหลดข้อมูลลงบอร์ด เมื่อโหลดโปรแกรมเสร็จแล้วขา 13 ของบอร์ดจะมีไฟกะพริบ



รูปที่ 5.23 โหลดโปรแกรมลงบอร์ด

5.7 ขั้นตอนการพัฒนาโปรแกรม

ขั้นตอนการพัฒนาโปรแกรมเริ่มจากเขียนโปรแกรมตามโครงสร้างของภาษาซี จากนั้นทำการคอมไพล์โปรแกรม หากเกิดข้อผิดพลาดต้องกลับไปแก้ไขโปรแกรมใหม่ ถ้าถูกต้องก็สามารถโหลดโปรแกรมลงบอร์ด Arduino และรันโปรแกรม หากผลการทำงานยังไม่สมบูรณ์ตามที่ต้องการก็กลับไปแก้ไขโปรแกรม



รูปที่ 5.24 ขั้นตอนการพัฒนาโปรแกรม

5.8 สรุป

ไมโครคอนโทรลเลอร์ คือตัวควบคุมขนาดเล็ก เป็นอุปกรณ์อิเล็กทรอนิกส์ชนิดหนึ่งซึ่งทำหน้าที่ประมวลผลตามโปรแกรมหรือชุดคำสั่งที่ป้อนเข้าไป โครงสร้างภายนอกมีลักษณะเป็นไอซี ส่วนโครงสร้างภายในจะเป็นวงจรรวมขนาดใหญ่ประกอบไปด้วย หน่วยคำนวณทางคณิตศาสตร์และลอจิก บัสข้อมูล บัสควบคุม บัสที่อยู่ พอร์ตขนาน พอร์ตอนุกรม รีจิสเตอร์ หน่วยความจำ วงจรนับ วงจรจับเวลา และวงจรอื่น ๆ รวมกันอยู่ภายใน ไมโครคอนโทรลเลอร์ถูกออกแบบมาเพื่อใช้ในงานควบคุม สามารถติดต่อกับอุปกรณ์อินพุตและเอาต์พุตได้สะดวก สามารถทำงานได้โดยใช้ชิปเดียว มีคำสั่งที่สนับสนุนการเขียนโปรแกรมควบคุม และสามารถเข้าถึงข้อมูลระดับบิตได้

Arduino เป็นไมโครคอนโทรเลอร์ขนาด 8 บิตในตระกูล AVR ที่ถูกออกแบบมาให้ใช้งานง่าย สามารถเชื่อมต่อกับคอมพิวเตอร์ด้วยพอร์ต USB มีบอร์ดอุปกรณ์เชื่อมต่อ หรือ Arduino Shield มากมายทำให้สะดวกในการพัฒนางาน อีกทั้งมีการเปิดเผยข้อมูลในการออกแบบและพัฒนาทั้งด้านฮาร์ดแวร์และซอฟต์แวร์ จึงมีผู้ใช้งานจำนวนมาก เหมาะสำหรับทุกคนที่สนใจเรียนรู้และต้องการประยุกต์ใช้งานไมโครคอนโทรลเลอร์

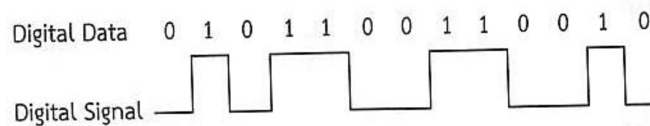
คำถามท้ายบทที่ 5

1. ไมโครคอนโทรลเลอร์คืออะไร
2. Arduino คืออะไรและมีข้อดีอย่างไร
3. จงเขียนขาสัญญาณของ Arduino Uno
4. ขาสัญญาณแอนะล็อกของ Arduino Uno มีขาสัญญาณใดบ้าง
5. ขาสัญญาณ PWM ของ Arduino Uno มีขาสัญญาณใดบ้าง
6. จงอธิบายวิธีการกำหนดค่าพอร์ตอนุกรมเพื่อการเชื่อมต่อกับบอร์ด Arduino
7. จงอธิบายการใช้งานเมนูคำสั่ง Verify, Upload และ New

6

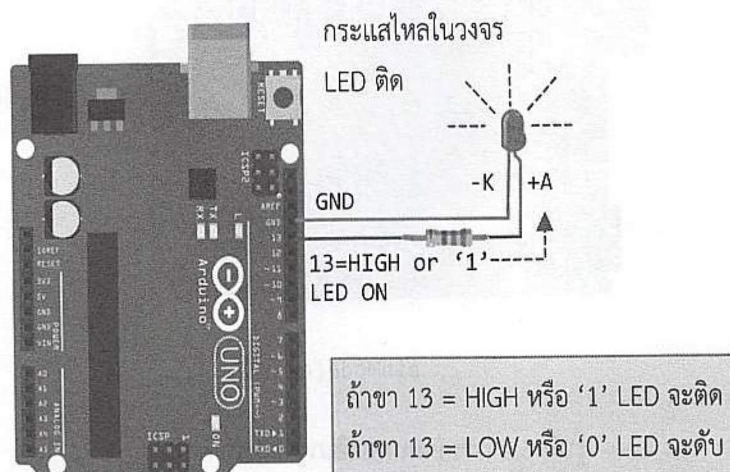
ดิจิทัลอินพุตและเอาต์พุต

ดิจิทัล (Digital) คือสัญญาณไฟฟ้าที่มี 2 ระดับ คือ สัญญาณลอจิก 0 และสัญญาณลอจิก 1 ซึ่งเป็นระบบตัวเลขฐานสอง สัญญาณลอจิก 0 แทนด้วยสัญญาณไฟ 0 โวลต์ และสัญญาณลอจิก 1 แทนด้วยสัญญาณไฟ 5 โวลต์ เพื่อใช้แทนความหมายของข้อมูลและระบบตัวเลขต่าง ๆ เพื่อให้วงจรดิจิทัลหรือคอมพิวเตอร์สามารถประมวลผลได้ตามการโปรแกรม



รูปที่ 6.1 สัญญาณดิจิทัล

6.1 การเชื่อมต่อกับหลอดแสดงผล LED



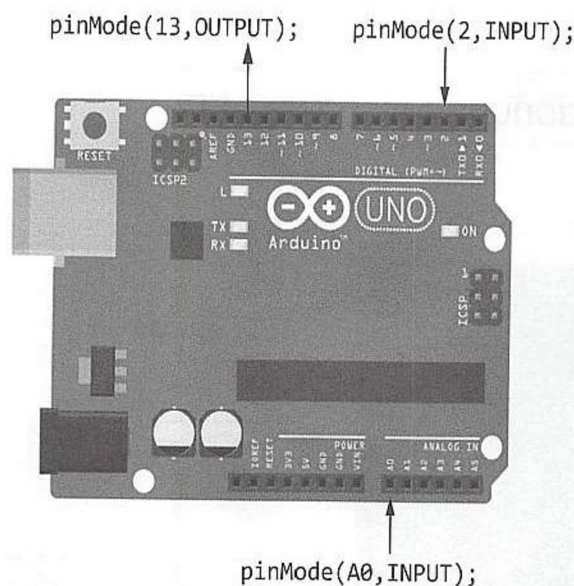
รูปที่ 6.2 วงจรการเชื่อมต่อ Arduino กับ LED

หลอดแสดงผล LED (Light Emitting Diode) หรือไดโอดเปล่งแสง เป็นอุปกรณ์อิเล็กทรอนิกส์ที่ใช้ในการแสดงผลเพื่อแสดงสถานะการทำงานของอุปกรณ์และไมโครคอนโทรลเลอร์ หลอด LED มี 2 ขา คือ ขาแอโนด (A) เป็นขาที่ต้องป้อนไฟบวก และขาแคโทด (K) เป็นขาไฟลบ ซึ่งต้องต่อให้ถูกต้อง การทำงานของวงจรคือเมื่อสัญญาณเอาต์พุตเป็นสัญญาณลอจิก 1 หรือสัญญาณไฟ 5 โวลต์จะมีกระแสไหลในวงจรทำให้หลอด LED ติด และเมื่อสัญญาณเอาต์พุตเป็นสัญญาณลอจิก 0 หรือสัญญาณไฟ 0 โวลต์จะไม่มีกระแสไหลในวงจร ส่วนตัวต้านทานจะทำหน้าที่ลดกระแสที่ไหลในวงจรให้เหมาะสมกับหลอด LED การเชื่อมต่อหลอด LED กับบอร์ด Arduino สามารถต่อได้ที่ขาดิจิทัลเอาต์พุตขา 2-13 ส่วนขา 0 และขา 1 ใช้ในการรับและส่งข้อมูลแบบอนุกรมระหว่างคอมพิวเตอร์กับบอร์ด Arduino

6.2 ฟังก์ชัน pinMode();

pinMode(); คือฟังก์ชันที่ใช้กำหนดขาสัญญาณของไมโครคอนโทรลเลอร์ให้มีสถานะเป็นอินพุตหรือเอาต์พุต

ตัวอย่างเช่น



รูปที่ 6.3 การกำหนดขาอินพุตและเอาต์พุต

pinMode(13,OUTPUT);

หมายถึง กำหนดให้ขา 13 เป็นขาเอาต์พุตเพื่อส่งสัญญาณออกไปควบคุมอุปกรณ์

pinMode(2,INPUT);

หมายถึง กำหนดให้ขา 2 เป็นขาอินพุตเพื่อรับข้อมูลจากสวิตช์หรือเซนเซอร์

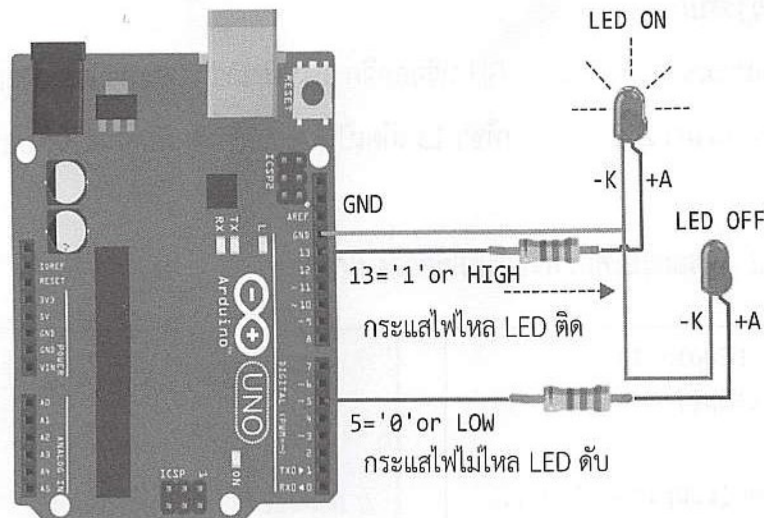
pinMode(A0,INPUT);

หมายถึง กำหนดให้ขา A0 เป็นขาอินพุตเพื่อรับข้อมูลจากสัญญาณแอนะล็อก

6.3 ฟังก์ชัน digitalWrite();

digitalWrite(); คือฟังก์ชันที่ใช้ส่งข้อมูลดิจิทัลออกขาเอาต์พุตเพื่อไปควบคุมการทำงานของอุปกรณ์

ตัวอย่างเช่น



รูปที่ 6.4 การส่งสัญญาณออกพอร์ตเอาต์พุต

digitalWrite(13,HIGH);

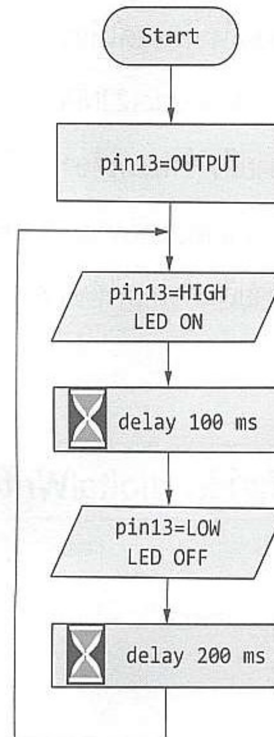
หมายถึง ให้เอาต์พุตขา 13 เป็นลอจิก 1 หรือมีสัญญาณไฟ 5 โวลต์

digitalWrite(5,LOW);

หมายถึง ให้เอาต์พุตขา 5 เป็นลอจิก 0 หรือมีสัญญาณไฟ 0 โวลต์

ตัวอย่างที่ 6.1 โปรแกรมไฟกะพริบ 1 ดวง

```
void setup()
{
  pinMode(13,OUTPUT);
}
void loop()
{
  digitalWrite(13,HIGH);
  delay(100);
  digitalWrite(13,LOW);
  delay(200);
}
```



รูปที่ 6.5 โฟลว์ชาร์ตไฟกะพริบขา 13

ผลการรันโปรแกรม

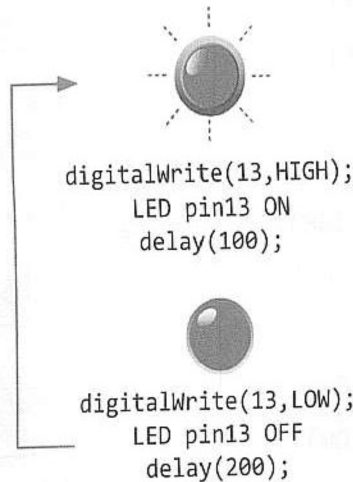
โปรแกรมจะส่งสัญญาณ HIGH หรือลอจิก 1 เป็นเวลา 100 ms และส่งสัญญาณ LOW หรือลอจิก 0 เป็นเวลา 200 ms ออกที่ขา 13 เกิดเป็นไฟกะพริบติดดับสลับกันที่บอร์ด Arduino

ตัวอย่างที่ 6.2 โปรแกรมไฟกะพริบแบบใช้คำสั่ง #define

```
#define LEDpin 13                  // กำหนดให้ LEDpin เป็น 13
void setup()
{
 pinMode(LEDpin,OUTPUT);           // กำหนดให้ขา 13 เป็น OUTPUT
}
void loop()                        // วนลูปตลอดกาล
{
 digitalWrite(LEDpin,HIGH);        // ขา 13 = HIGH ไฟ LED ติด
 delay(100);                       // หน่วงเวลา 100 ms
 digitalWrite(LEDpin,LOW);         // ขา 13 = LOW ไฟ LED ดับ
 delay(200);                       // หน่วงเวลา 200 ms
}
```

ผลการรันโปรแกรม

โปรแกรมจะส่งสัญญาณ HIGH หรือลอจิก 1 เป็นเวลา 100 ms และส่งสัญญาณ LOW หรือลอจิก 0 เป็นเวลา 200 ms ออกที่ขา 13 เกิดเป็นไฟกะพริบติดดับสลับกัน โดยกำหนดให้ LEDpin เป็นขา 13



รูปที่ 6.6 ไฟกะพริบขา 13

ตัวอย่างที่ 6.3 โปรแกรมไฟกะพริบแบบประกาศตัวแปร

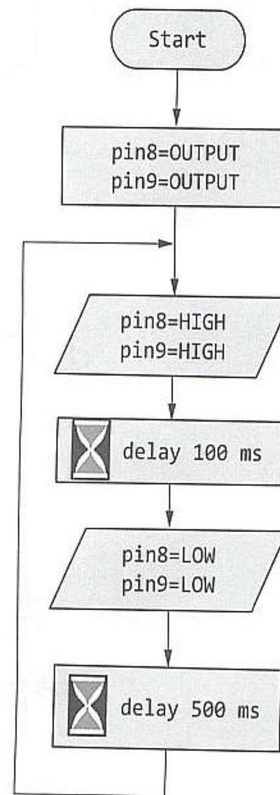
```
int LEDpin=13;                    // ประกาศตัวแปร LEDpin เท่ากับ 13
void setup()
{
  pinMode(LEDpin,OUTPUT);         // กำหนดให้ขา 13 เป็น OUTPUT
}
void loop()                       // วนลูปตลอดกาล
{
  digitalWrite(LEDpin,HIGH);      // ขา 13 = HIGH ไฟ LED ติด
  delay(100);                     // หน่วงเวลา 100 ms
  digitalWrite(LEDpin,LOW);       // ขา 13 = LOW ไฟ LED ดับ
  delay(200);                     // หน่วงเวลา 200 ms
}
```

ผลการรันโปรแกรม

โปรแกรมจะส่งสัญญาณ HIGH หรือลอจิก 1 เป็นเวลา 100 ms และส่งสัญญาณ LOW หรือลอจิก 0 เป็นเวลา 200 ms ออกที่ขา 13 เกิดเป็นไฟกะพริบติดดับสลับกัน โดยการประกาศตัวแปร LEDpin ให้เท่ากับขา 13 ของไมโครคอนโทรลเลอร์

ตัวอย่างที่ 6.4 โปรแกรมไฟกะพริบ 2 ดวง

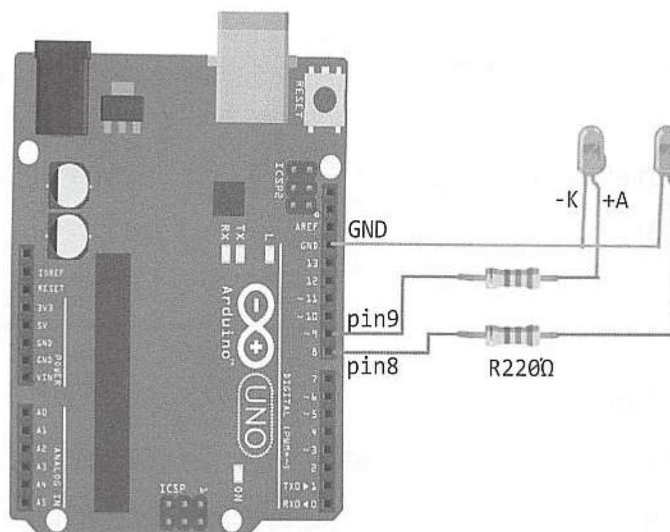
```
void setup()
{
   pinMode(8,OUTPUT);
   pinMode(9,OUTPUT);
}
void loop()
{  digitalWrite(8,HIGH);
   digitalWrite(9,HIGH);
   delay(100);
   digitalWrite(8,LOW);
   digitalWrite(9,LOW);
   delay(500);
}
```



รูปที่ 6.7 โฟลว์ชาร์ตไฟกะพริบ 2 ดวง

ผลการรันโปรแกรม

โดยเมื่อต่อหลอด LED เข้าที่ขา 8 และ 9 ของบอร์ด Arduino โปรแกรมจะส่งสัญญาณ HIGH หรือลอจิก 1 เป็นเวลา 100 ms และส่งสัญญาณ LOW หรือลอจิก 0 เป็นเวลา 500 ms ออกที่ขา 8 และขา 9 เกิดเป็นไฟกะพริบติดดับพร้อมกัน 2 ดวง



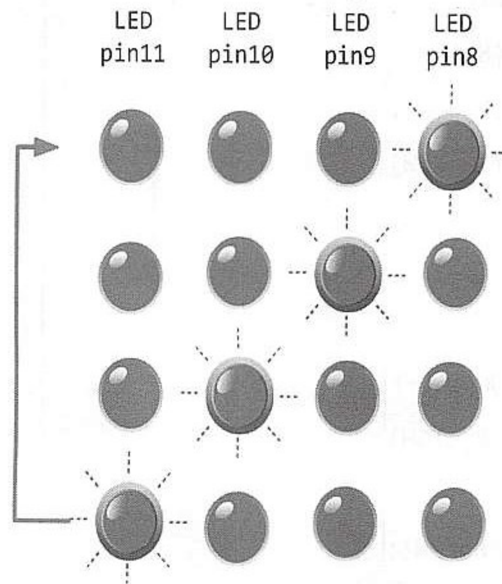
รูปที่ 6.8 วงจรไฟกะพริบ 2 ดวง

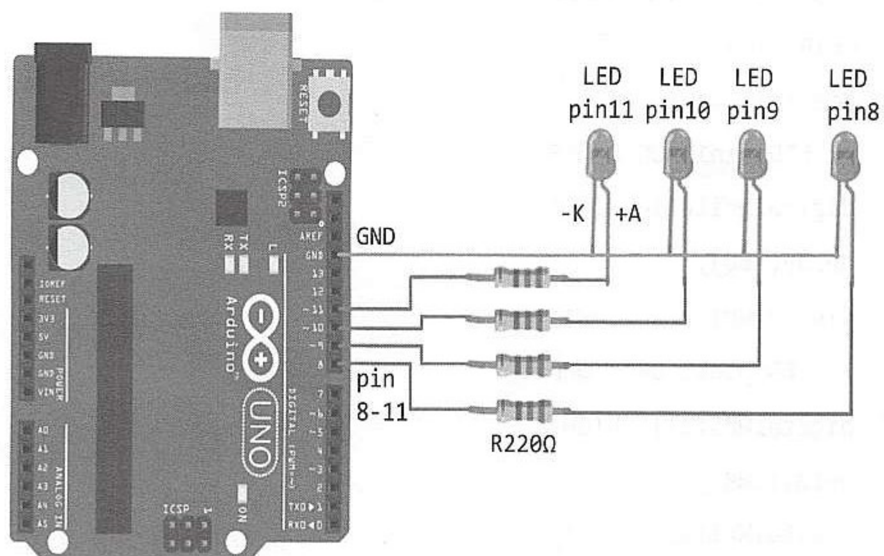
ตัวอย่างที่ 6.5 โปรแกรมไฟวิ่ง 4 ดวง

```
void setup()
{ pinMode(8,OUTPUT); pinMode(9,OUTPUT);
  pinMode(10,OUTPUT); pinMode(11,OUTPUT);
  digitalWrite(8,LOW);
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
  digitalWrite(11,LOW);
}
void loop()
{ // LED pin8 ON & OFF
  digitalWrite(8,HIGH);
  delay(200);
  digitalWrite(8,LOW);
  // LED pin9 ON & OFF
  digitalWrite(9,HIGH);
  delay(200);
  digitalWrite(9,LOW);
  // LED pin10 ON & OFF
  digitalWrite(10,HIGH);
  delay(200);
  digitalWrite(10,LOW);
  // LED pin11 ON & OFF
  digitalWrite(11,HIGH);
  delay(200);
  digitalWrite(11,LOW);
}
```

ผลการรันโปรแกรม

โปรแกรมจะส่งสัญญาณ HIGH หรือลอจิก 1 แล้วหน่วงเวลา 200 ms จากนั้นส่งสัญญาณ LOW หรือลอจิก 0 โดยไล่ไปทีละขาจากขา 8 ถึง 11 แล้ววนกลับมาใหม่ ทำให้เกิดเป็นไฟวิ่ง 4 ดวง

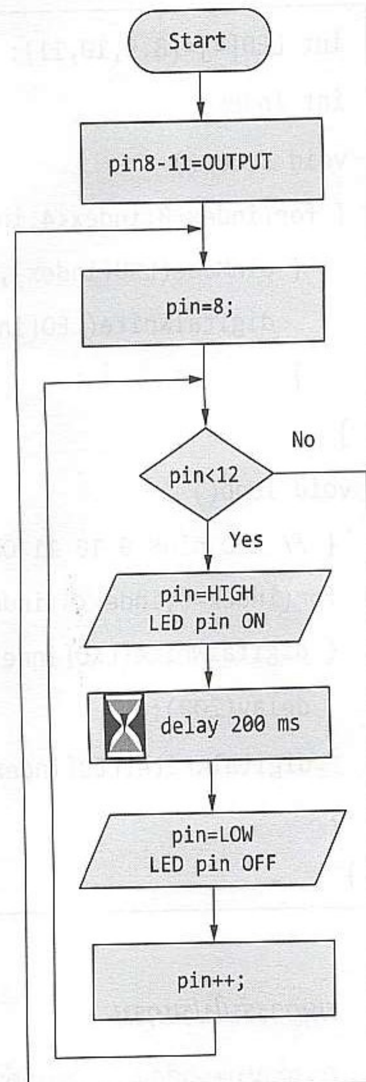




รูปที่ 6.9 วงจรไฟวิ่ง 4 ดวง

ตัวอย่างที่ 6.6 โปรแกรมไฟวิ่ง 4 ดวงแบบวนรอบโดยใช้ฟังก์ชัน for()

```
int pin;
void setup()
{
 for(pin=8;pin<12;pin++)
  { pinMode(pin,OUTPUT);
    digitalWrite(pin,LOW);
  }
}
void loop()
{ // LED pin8 9 10 11 ON & OFF
 for(pin=8;pin<12;pin++)
  {
   digitalWrite(pin,HIGH);
   delay(200);
   digitalWrite(pin,LOW);
  }
}
```



รูปที่ 6.10 โฟลว์ชาร์ตไฟวิ่ง 4 ดวงแบบวนรอบ

ผลการรันโปรแกรม

การกำหนดขาสัญญาณและการส่งข้อมูลออกขาเอาต์พุตจะใช้การวนรอบ โดยโปรแกรมจะส่งสัญญาณ HIGH หรือลอจิก 1 แล้วหน่วงเวลา 200 ms จากนั้นส่งสัญญาณ LOW หรือลอจิก 0 ออกที่ขา 8 ถึง 11 ทำให้เป็นไฟวิ่ง 4 ดวง

ตัวอย่างที่ 6.7 โปรแกรมไฟวิ่ง 4 ดวงแบบอาร์เรย์

```
int LED[4]={8,9,10,11};
int index;
void setup()
{ for(index=0;index<4;index++)
   { pinMode(LED[index],OUTPUT);
     digitalWrite(LED[index],LOW);
   }
}
void loop()
 { // LED pin8 9 10 11 ON & OFF
 for(index=0;index<4;index++)
 { digitalWrite(LED[index],HIGH);
   delay(200);
   digitalWrite(LED[index],LOW);
 }
}
```

ผลการรันโปรแกรม

การกำหนดขาสัญญาณจะถูกกำหนดไว้ในตัวแปรอาร์เรย์ LED[4] และการส่งข้อมูลออกขาเอาต์พุตจะใช้การวนรอบ โดยโปรแกรมจะส่งสัญญาณ HIGH หรือลอจิก 1 แล้วหน่วงเวลา 200 ms จากนั้นส่งสัญญาณ LOW หรือลอจิก 0 ออกที่ขา 8 ถึง 11 ทำให้เป็นไฟวิ่ง 4 ดวง

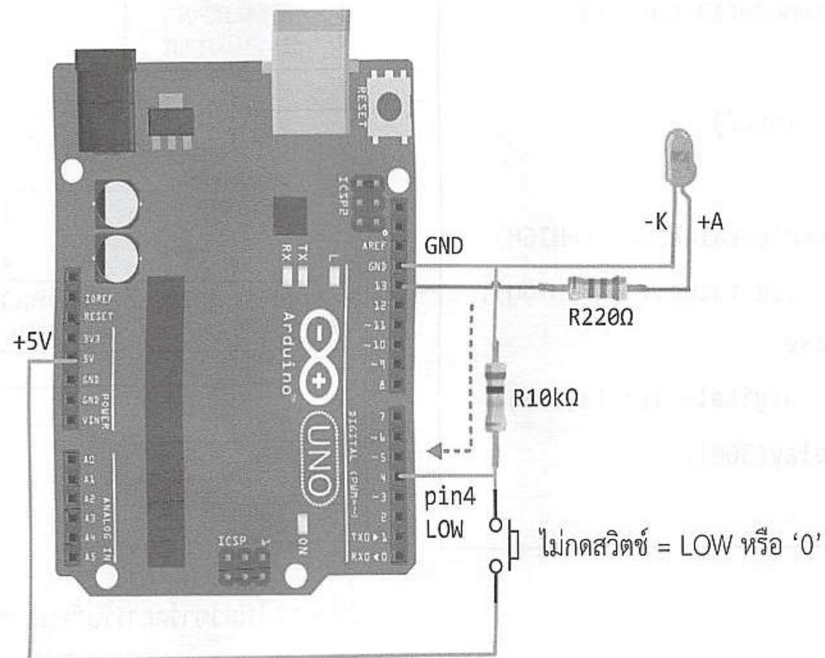
6.4 ฟังก์ชัน digitalRead();

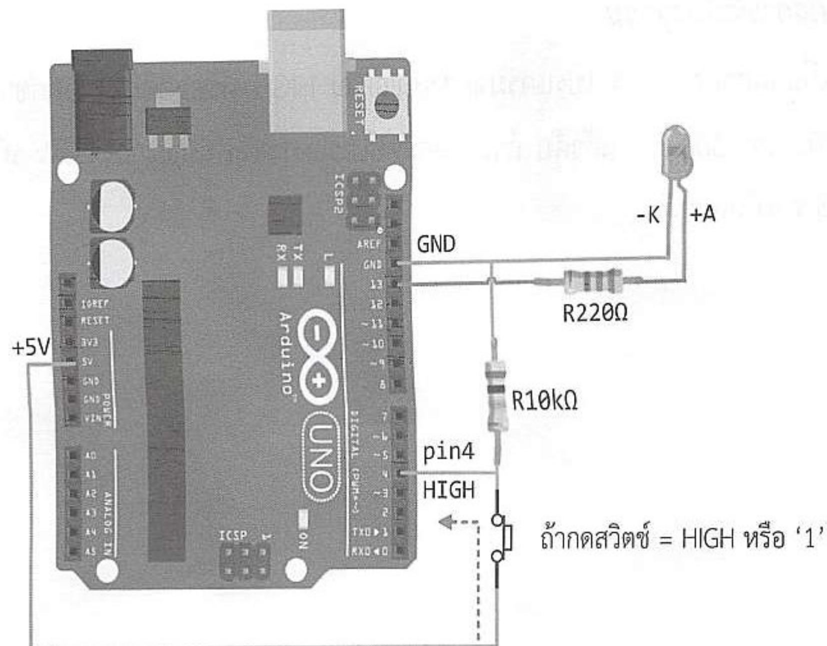
digitalRead(); คือฟังก์ชันที่ใช้อ่านค่าข้อมูลดิจิทัลของขาสัญญาณไมโครคอนโทรลเลอร์ซึ่งสามารถอ่านได้ทั้งขาอินพุตและเอาต์พุต

ตัวอย่างเช่น

s1=digitalRead(4); หมายถึง อ่านค่าข้อมูลขา 4 ว่าเป็นลอจิก 0 หรือลอจิก 1

if(digitalRead(4)==HIGH) หมายถึง ถ้าค่าข้อมูลขา 4 เป็น HIGH หรือลอจิก 1 ให้ทำคำสั่งในฟังก์ชัน if()

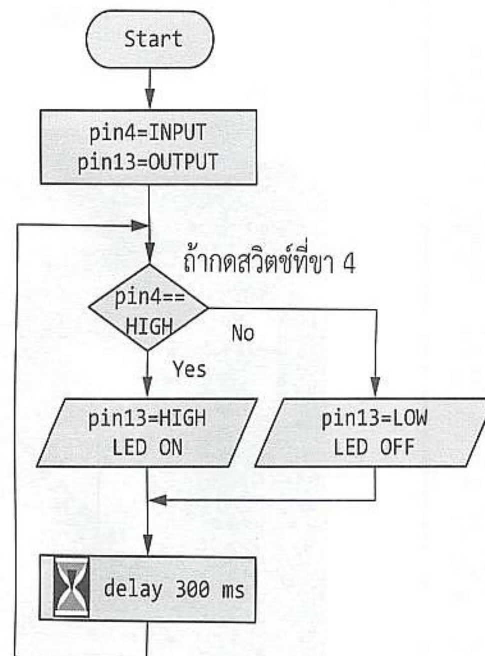




รูปที่ 6.11 การทำงานของวงจรสวิตช์

ตัวอย่างที่ 6.8 โปรแกรมการรับข้อมูลจากสวิตช์

```
void setup()
{
  pinMode(4,INPUT);
  pinMode(13,OUTPUT);
}
void loop()
{
  if(digitalRead(4)==HIGH)
     digitalWrite(13,HIGH);
  else
     digitalWrite(13,LOW);
  delay(300);
}
```

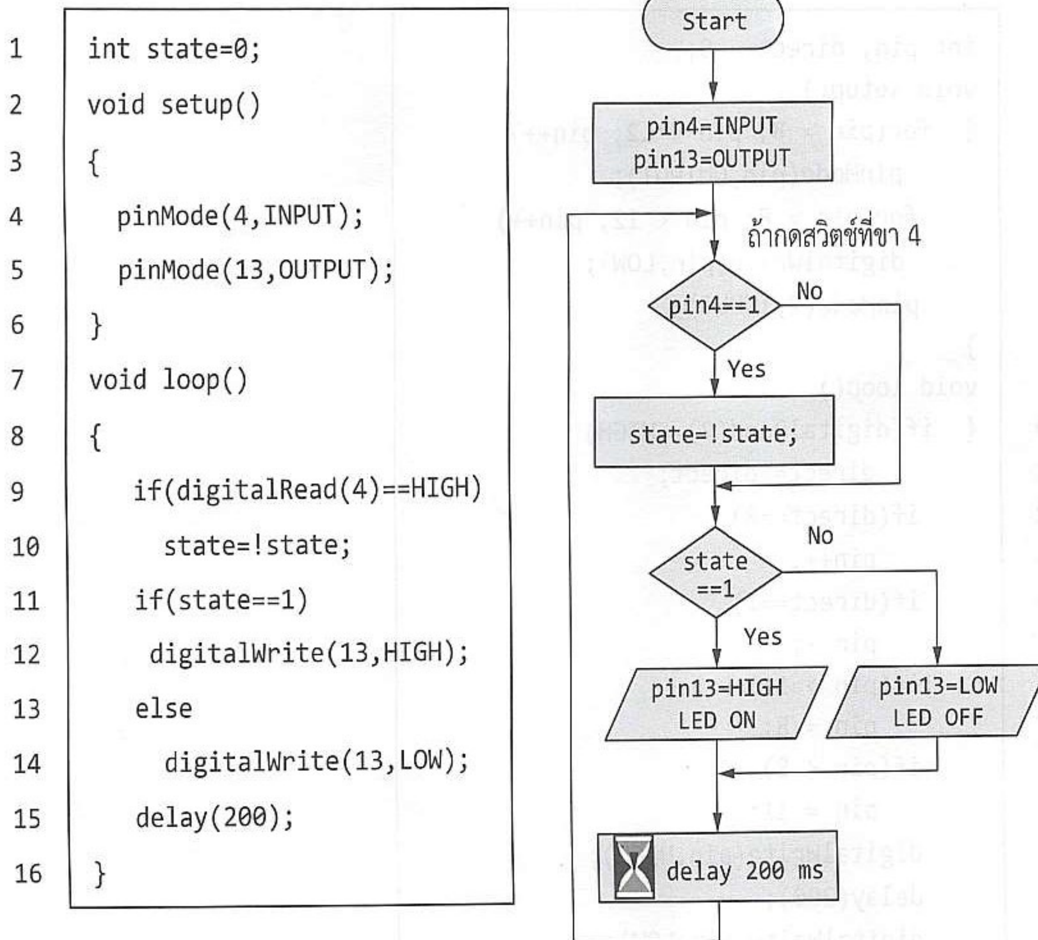


รูปที่ 6.12 โฟลว์ชาร์ตการรับข้อมูลจากสวิตช์

ผลการรันโปรแกรม

เมื่อกดสวิตช์ที่ขา 4 โปรแกรมจะส่งสัญญาณ HIGH หรือลอจิก 1 ออกขา 13 ทำให้หลอดติดเป็นเวลา 300 ms แล้วดับ ถ้าไม่กดสวิตช์โปรแกรมจะส่งสัญญาณ LOW หรือลอจิก 0 ออกขา 13 ทำให้หลอดดับ

ตัวอย่างที่ 6.9 โปรแกรมการรับข้อมูลจากสวิตช์แบบ Toggle



```
int state=0;
void setup()
{
  pinMode(4,INPUT);
  pinMode(13,OUTPUT);
}
void loop()
{
   if(digitalRead(4)==HIGH)
     state=!state;
   if(state==1)
    digitalWrite(13,HIGH);
   else
     digitalWrite(13,LOW);
   delay(200);
}
```


รูปที่ 6.13 โฟลว์ชาร์ตการรับข้อมูลจากสวิตช์แบบ Toggle

ผลการรันโปรแกรม

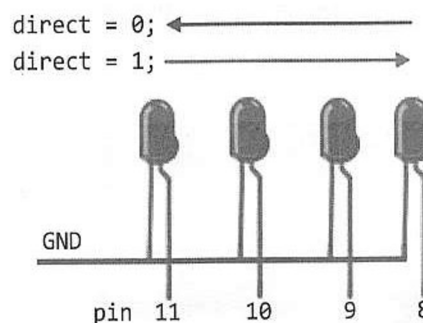
เมื่อกดสวิตช์ที่ขา 4 จะทำให้ตัวแปร state กลับสถานะจากลอจิก 0 เป็นลอจิก 1 หรือจากลอจิก 1 เป็นลอจิก 0 จากนั้นทำการเปรียบเทียบ ถ้าตัวแปร state เท่ากับ 1 จะส่งสัญญาณ HIGH หรือลอจิก 1 ออกขา 13 ทำให้หลอดติด ถ้าไม่ใช่จะส่งสัญญาณ LOW หรือลอจิก 0 ออกขา 13 ทำให้หลอดดับ ดังนั้นหลอดจะเปลี่ยนสถานะการทำงานเมื่อมีการกดสวิตช์

ตัวอย่างที่ 6.10 โปรแกรมไฟวิ่งซ้ายขวาตามการกดสวิตช์

```
int pin, direct = 0;
void setup()
{  for(pin = 8; pin < 12; pin++)
      pinMode(pin,OUTPUT);
     for(pin = 8; pin < 12; pin++)
      digitalWrite(pin,LOW);
     pinMode(2,INPUT);
}
void loop()
{  if(digitalRead(2)==HIGH)
        direct=!direct;
     if(direct==0)
        pin++;
     if(direct==1)
        pin--;
     if(pin > 11)
        pin = 8;
     if(pin < 8)
        pin = 11;
     digitalWrite(pin,HIGH);
     delay(200);
     digitalWrite(pin,LOW);
}
```

ผลการรันโปรแกรม

เมื่อกดสวิตช์ที่ขา 2 จะทำให้ตัวแปร direct กลับสถานะจากลอจิก 0 เป็นลอจิก 1 หรือจากลอจิก 1 เป็นลอจิก 0 จากนั้นทำการเปรียบเทียบ ถ้าตัวแปร direct เท่ากับ 0 จะให้ไฟวิ่งจากขวาไปซ้าย แต่ถ้าตัวแปร direct เท่ากับ 1 จะให้ไฟวิ่งจากซ้ายไปขวา



รูปที่ 6.14 ไฟวิ่งซ้ายขวาตามการกดสวิตช์

6.5 ฟังก์ชัน millis();

millis(); เป็นฟังก์ชันเกี่ยวกับเวลาในการรันโปรแกรม มีหน่วยเป็นมิลลิวินาที โดยจะเก็บค่าได้สูงสุดเป็น unsigned long ขนาด 32 บิต การใช้ฟังก์ชัน millis(); แทน delay(); เพื่อไม่ให้ซีพียูวนรอบติดอยู่ในลูปของคำสั่ง delay(); จนไม่สามารถทำงานอื่นได้

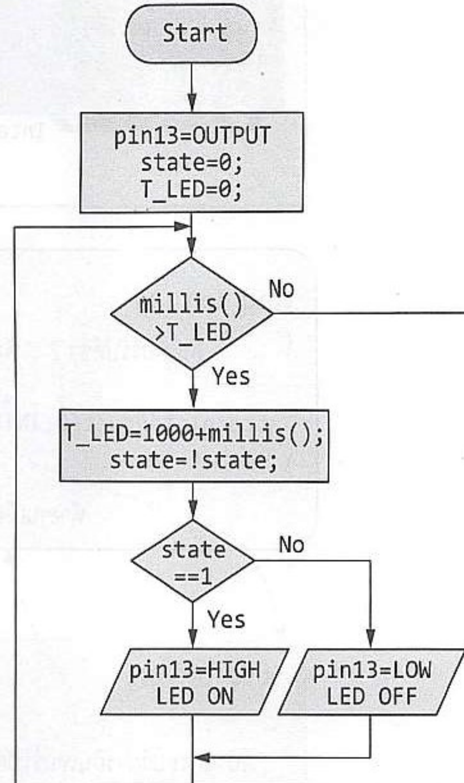
ตัวอย่างเช่น

```
if(millis()>next_T)
    next_T=1000+millis();
```

หมายถึง ถ้า millis() มีค่ามากกว่า next_T ให้ next_T=1000+millis(); โดยทุก ๆ 1000 มิลลิวินาที หรือ 1 วินาที เงื่อนไขในฟังก์ชัน if() จะเป็นจริง และฟังก์ชัน millis(); จะเพิ่มค่าเวลาอัตโนมัติตามการรันโปรแกรม

ตัวอย่างที่ 6.11 โปรแกรมไฟกะพริบโดยใช้ฟังก์ชัน millis();

```
unsigned long T_LED;
int state;
void setup()
{
  pinMode(13,OUTPUT);
}
void loop()
{  if(millis()>T_LED)
   { T_LED=1000+millis();
     state=!state;
     if(state==1)
       digitalWrite(13,HIGH);
     else
       digitalWrite(13,LOW);
   }
}
```



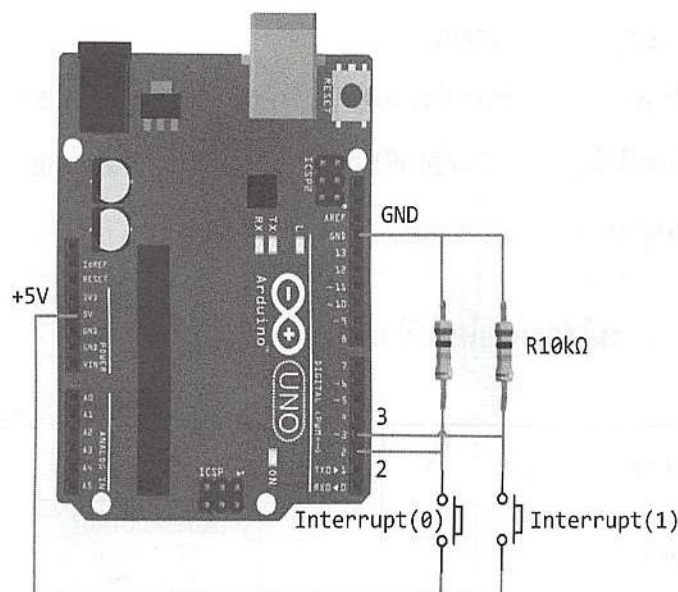
รูปที่ 6.15 โฟลว์ชาร์ตไฟกะพริบโดยใช้ฟังก์ชัน millis();

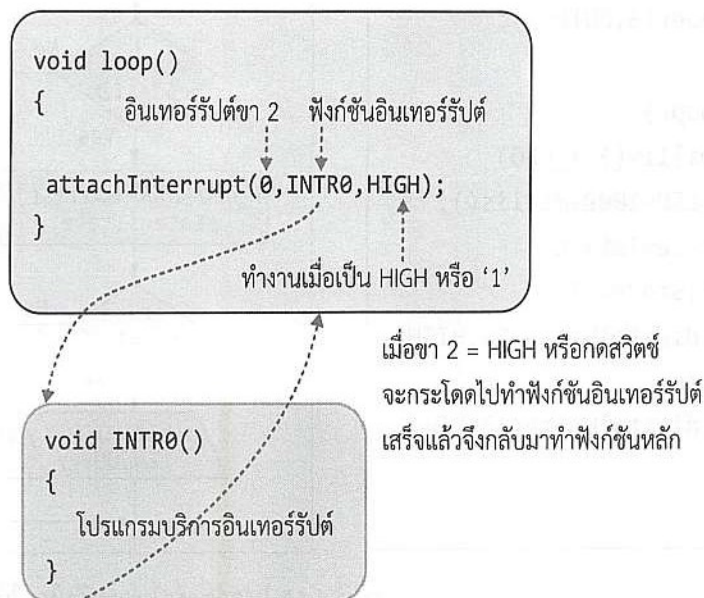
ผลการรันโปรแกรม

ทุก ๆ 1000 ms โปรแกรมจะส่งสัญญาณ HIGH และสัญญาณ LOW สลับกัน เกิดเป็นไฟกะพริบที่ขา 13 โดยค่าของ millis() จะเพิ่มค่าอัตโนมัติตามการรันโปรแกรม

6.6 อินเทอร์รัปต์

อินเทอร์รัปต์ (Interrupt) หมายถึง การขัดจังหวะการทำงานของไมโครคอนโทรลเลอร์ที่กำลังประมวลผลอยู่ในโปรแกรมหลัก เมื่อไมโครคอนโทรลเลอร์ได้รับสัญญาณอินเทอร์รัปต์จะหยุดการประมวลผลโปรแกรมหลักชั่วคราว แล้วกระโดดไปทำงานที่โปรแกรมบริการอินเทอร์รัปต์จนแล้วเสร็จ จึงกระโดดกลับไปทำงานที่โปรแกรมหลักต่อ โดยไมโครคอนโทรลเลอร์ Arduino UNO R3 มีสัญญาณอินเทอร์รัปต์ภายนอกที่ขา 2 และขา 3





รูปที่ 6.16 วงจรอินเทอร์รัปต์และการทำงาน

ตัวอย่างที่ 6.12 การเขียนโปรแกรม Interrupt(0) หรืออินเทอร์รัปต์ที่ขา 2

```
volatile int I;
void setup()
 {
  attachInterrupt(0,INTR,HIGH);
  pinMode(13,OUTPUT);
 }
void loop()
 {
  if(I==1)
  { digitalWrite(13,HIGH);
    delay(100);
    digitalWrite(13,LOW);
    delay(200);
    I=0;
  }
 }
void INTR()
 { I=1;
 }
```

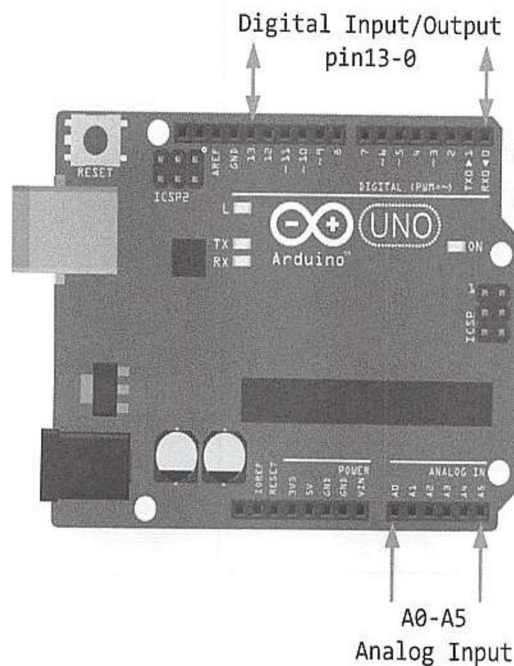
ผลการรันโปรแกรม

เมื่อกดสวิตช์อินเทอร์รัปต์ที่ขา 2 โปรแกรมจะกระโดดไปยังฟังก์ชันอินเทอร์รัปต์ กำหนดให้ตัวแปร I เท่ากับ 1 แล้วกระโดดมายังโปรแกรมหลัก ทำให้เกิดไฟกะพริบ 1 ครั้งเมื่อกดสวิตช์อินเทอร์รัปต์

volatile เป็นตัวแปรชนิดหนึ่งเพื่อบอกให้คอมไพเลอร์ (Compiler) รับรู้ว่าตัวแปรชนิดนี้มีการเปลี่ยนแปลงค่าได้จากนอกฟังก์ชัน void loop() ซึ่งใช้กับการเขียนโปรแกรมอินเทอร์รัปต์

6.7 สรุป

ในการเขียนโปรแกรมจะต้องมีการกำหนดขาสัญญาณให้เป็นอินพุตหรือเอาต์พุตโดยใช้ฟังก์ชัน pinMode(); และถ้าต้องการส่งข้อมูลดิจิทัลออกต้องใช้ฟังก์ชัน digitalWrite(); ส่วนการรับข้อมูลเข้าจะใช้ฟังก์ชัน digitalRead();



รูปที่ 6.17 ขาสัญญาณอินพุตและเอาต์พุต

ตัวอย่างเช่น

pinMode(13,OUTPUT);

หมายถึง กำหนดให้ขา 13 เป็นขาเอาต์พุตเพื่อส่งสัญญาณออกไปควบคุมอุปกรณ์

pinMode(2,INPUT);

หมายถึง กำหนดให้ขา 2 เป็นขาอินพุตเพื่อรับข้อมูลจากสวิตช์หรือเซนเซอร์

digitalWrite(13,HIGH);

หมายถึง ให้เอาต์พุตขา 13 เป็นลอจิก 1 หรือมีสัญญาณไฟ 5 โวลต์

if(digitalRead(2)==HIGH)

หมายถึง ถ้าค่าข้อมูลขา 2 เป็น HIGH หรือลอจิก 1 ให้ทำคำสั่งในฟังก์ชัน if()

อินเทอร์รัปต์หมายถึงการขัดจังหวะการทำงาน เมื่อได้รับสัญญาณอินเทอร์รัปต์ ไมโครคอนโทรลเลอร์จะหยุดการประมวลผลที่โปรแกรมหลักชั่วคราว แล้วกระโดดไปทำงานที่โปรแกรมบริการอินเทอร์รัปต์จนแล้วเสร็จ จึงกระโดดกลับไปทำงานที่โปรแกรมหลักต่อ โดยไมโครคอนโทรลเลอร์ Arduino UNO R3 มีสัญญาณอินเทอร์รัปต์ภายนอกที่ขา 2 และขา 3

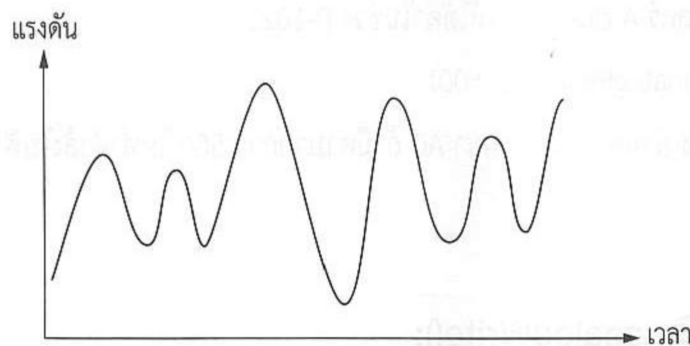
คำถามท้ายบทที่ 6

1. จงอธิบายฟังก์ชัน pinMode(); พร้อมยกตัวอย่าง
2. จงอธิบายฟังก์ชัน digitalWrite(); พร้อมยกตัวอย่าง
3. จงอธิบายฟังก์ชัน digitalRead(); พร้อมยกตัวอย่าง
4. จงอธิบายหลักการอินเทอร์รัปต์
5. จงเขียนโปรแกรมไฟกะพริบที่ขา 8 ให้กะพริบทุก 1000 ms และที่ขา 9 ให้กะพริบทุก 500 ms โดยใช้ฟังก์ชัน millis();
6. จงเขียนโปรแกรมรับสัญญาณอินเทอร์รัปต์ที่ขา 2 และขา 3

7

แอนะล็อกอินพุตและเอาต์พุต

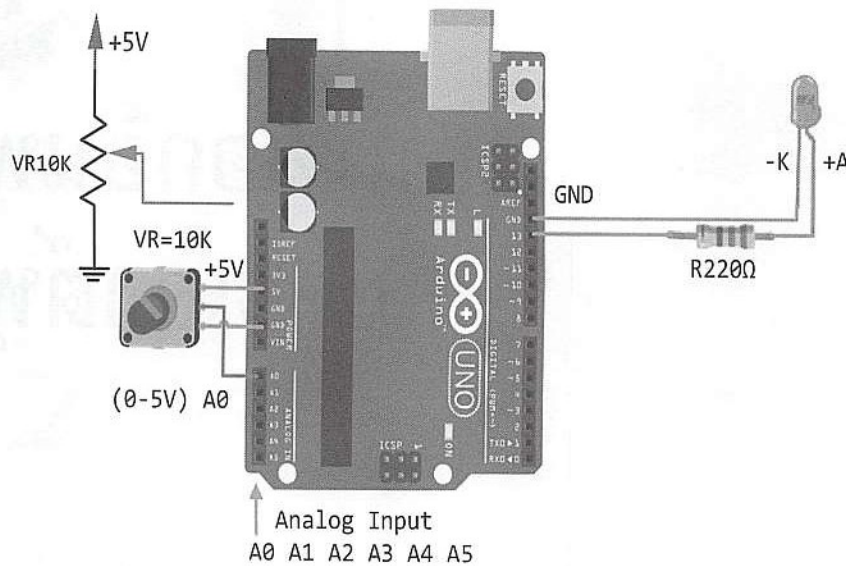
แอนะล็อก (Analog) คือสัญญาณที่มีความต่อเนื่อง มีหลายค่า เป็นสัญญาณธรรมชาติ ซึ่งต่างจากสัญญาณดิจิทัลที่มีเพียง 2 ระดับ สัญญาณแอนะล็อกมีลักษณะเป็นคลื่นที่มีความถี่และขนาดหรือความเข้มของสัญญาณที่ต่างกันและมักเปลี่ยนแปลงตลอดเวลา มีทั้งสัญญาณที่มีการเปลี่ยนแปลงเป็นรูปแบบแน่นอน เช่น สัญญาณไซน์ (Sine Wave) หรือสัญญาณไฟฟ้ากระแสสลับ และยังมีสัญญาณแอนะล็อกที่เกิดขึ้นโดยธรรมชาติที่มีลักษณะแตกต่างกัน เช่น สัญญาณเสียง แสง ความชื้น อุณหภูมิ



รูปที่ 7.1 สัญญาณแอนะล็อก

7.1 ฟังก์ชัน analogRead();

analogRead(); คือฟังก์ชันอ่านค่าข้อมูลแอนะล็อกจากขา A0-A5 ของไมโครคอนโทรลเลอร์ โดยผ่านวงจรแปลงสัญญาณแอนะล็อก 0-5 โวลต์ เป็นสัญญาณดิจิทัลมีค่าอยู่ระหว่าง 0-1023 มีความละเอียดขนาด 10 บิต (A/D 10 bits) ซึ่งเป็นวงจรที่มีอยู่ภายในไมโครคอนโทรลเลอร์



รูปที่ 7.2 วงจรการเชื่อมต่อตัวต้านทานแบบปรับค่าได้

ตัวอย่างเช่น

```
int VR=analogRead(A0);
```

หมายถึง อ่านค่าข้อมูลจากขาแอนะล็อก A0 มาเก็บไว้ในตัวแปร VR ซึ่งเป็นตัวแปรชนิดเลขจำนวนเต็ม 16 บิต โดยสัญญาณจากค่าความต้านทานแบบปรับค่าได้ (Variable Resistor : VR) มีค่า 0-5 โวลต์ ผ่านวงจรแปลงสัญญาณแอนะล็อกเป็นดิจิทัลขนาด 10 บิตที่อยู่ภายในไมโครคอนโทรลเลอร์ Arduino ทำให้ได้ค่าในช่วง 0-1023

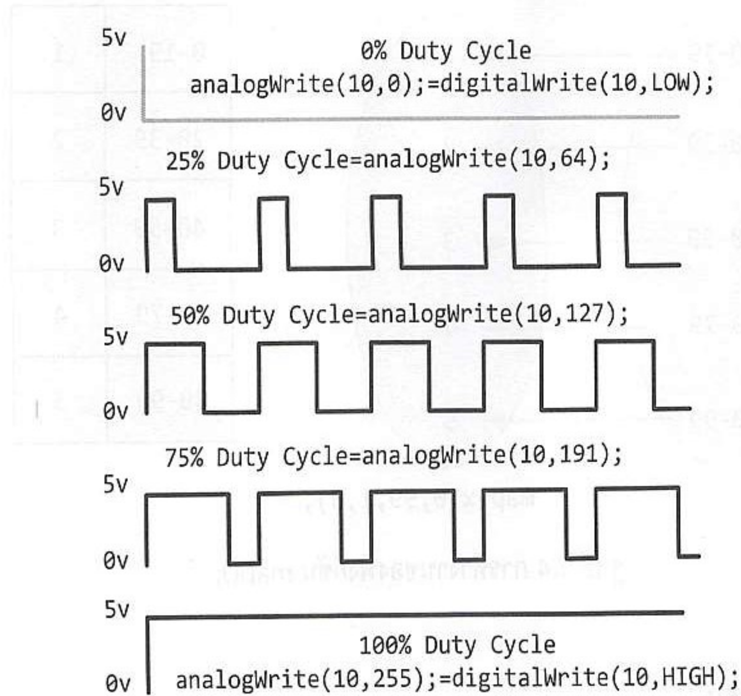
```
if(analogRead(A0)>500)
```

หมายถึง อ่านค่าข้อมูลจากขา A0 ถ้ามีค่ามากกว่า 500 ให้ทำคำสั่งในฟังก์ชัน if()

7.2 ฟังก์ชัน analogWrite();

analogWrite(); คือฟังก์ชันส่งข้อมูลแอนะล็อกออกขาเอาต์พุต ซึ่งเป็นการส่งแบบ PWM (Pulse Width Modulation) และค่าที่ส่งออกอยู่ระหว่าง 0-255 ในบอร์ด Arduino UNO จะมีขาเอาต์พุต PWM อยู่ 6 ขา คือขา 3, 5, 6, 9, 10 และ 11

สัญญาณ PWM คือสัญญาณที่มีการปรับความกว้างของสัญญาณพัลส์ตามสัดส่วนที่กำหนด



รูปที่ 7.3 สัญญาณ PWM

ตัวอย่างเช่น

analogWrite(10,127);

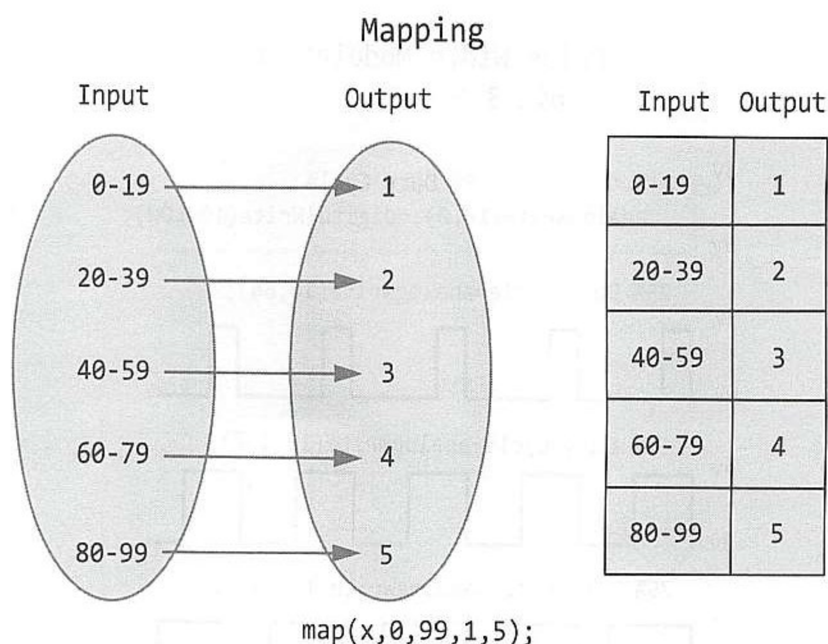
หมายถึง ให้ส่งสัญญาณพัลส์ที่มีความกว้างของสัญญาณพัลส์ 50% ออกที่ขา 10

analogWrite(10,255);

หมายถึง ให้ส่งสัญญาณพัลส์ที่มีความกว้างของสัญญาณพัลส์ 100% ซึ่งเป็นค่าสูงสุดออกที่ขา 10

7.3 ฟังก์ชัน map();

map(); เป็นฟังก์ชันหรือคำสั่งที่ใช้ในการแปลงค่าจากช่วงตัวเลขจำนวนหนึ่งให้เป็นช่วงตัวเลขอีกจำนวนหนึ่ง เพื่อความเหมาะสมในการนำไปใช้งาน



รูปที่ 7.4 การทำงานของฟังก์ชัน map();

ตัวอย่างเช่น

x=map(x,0,99,1,5);

หมายถึง การแปลงค่าของตัวแปร x จาก 0-99 ให้อยู่ในช่วง 1-5 แล้วเก็บไว้ในตัวแปร x

VR=map(VR,0,1023,0,100);

หมายถึง การแปลงค่าของตัวแปร VR จาก 0-1023 ให้อยู่ในช่วง 0-100 แล้วเก็บไว้ในตัวแปร VR ซึ่งช่วงตัวเลข 0-100 สามารถเข้าใจได้ง่ายกว่าช่วงตัวเลข 0-1023

7.4 ฟังก์ชัน Serial

Serial เป็นฟังก์ชันเกี่ยวกับการรับและส่งข้อมูลแบบอนุกรม

Serial.begin(); กำหนดอัตราการรับและส่งข้อมูลแบบอนุกรม มีหน่วยเป็นบิตต่อวินาที เช่น 9600 หรือ 14400 บิตต่อวินาที

Serial.print(); เป็นฟังก์ชันที่ใช้ในการแสดงผลหรือพิมพ์ข้อมูลที่รับมาจากบอร์ด Arduino โดยการรับและส่งข้อมูลแบบอนุกรม

Serial.println(); เป็นฟังก์ชันที่ใช้ในการแสดงผลหรือพิมพ์ข้อมูลและมีการขึ้นบรรทัดใหม่ทุกครั้ง

ตัวอย่างเช่น

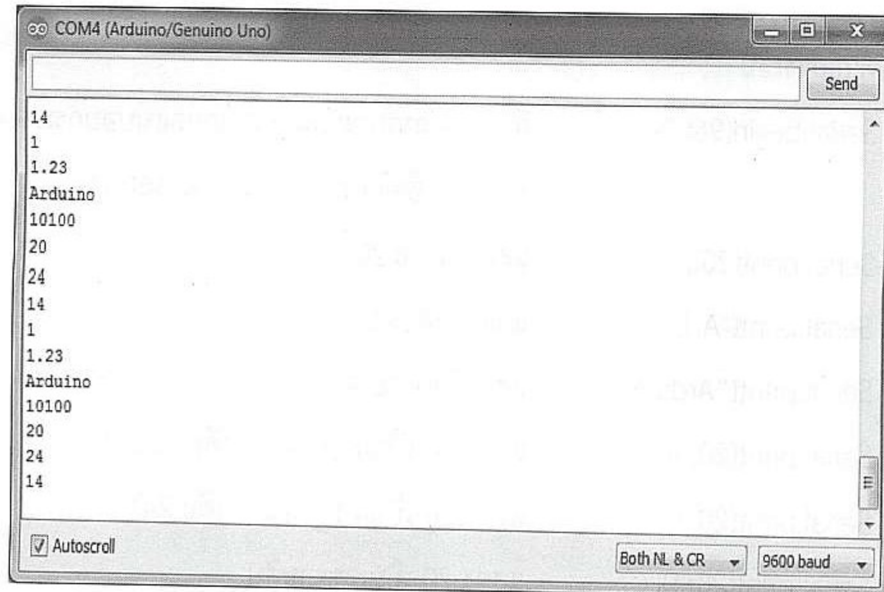
Serial.begin(9600);	กำหนดอัตราการรับและส่งข้อมูลแบบอนุกรม 9600 บิตต่อวินาที ซึ่งต้องกำหนดใน void setup()
Serial.print(20);	แสดงตัวเลข 20
Serial.print('A');	แสดงตัวอักษร A
Serial.print("Arduino");	แสดงข้อความ Arduino
Serial.print(20, BIN);	แสดง 20 เป็นเลขฐานสอง (คือ 10100)
Serial.print(20, OCT);	แสดง 20 เป็นเลขฐานแปด (คือ 24)
Serial.print(20, DEC);	แสดง 20 เป็นเลขฐานสิบ
Serial.print(20, HEX);	แสดง 20 เป็นเลขฐานสิบหก (คือ 14)
Serial.println(1.23456, 2);	แสดง 1.23456 เป็นเลขทศนิยม 2 ตำแหน่ง (คือ 1.23) และขึ้นบรรทัดใหม่

ตัวอย่างที่ 7.1 โปรแกรมแสดงข้อมูลผ่าน Serial Monitor

```
int a=20;
void setup()
{  Serial.begin(9600);
}
void loop()
{  Serial.println(1);
   Serial.println(1.234);
   Serial.println("Arduino");
   Serial.println(a,BIN);
   Serial.println(a,DEC);
   Serial.println(a,OCT);
   Serial.println(a,HEX);
 delay(1000);
}
```

ผลการรันโปรแกรม

ให้ไปที่เมนูคำสั่ง Tools แล้วเลือกคำสั่ง Serial Monitor โปรแกรมจะแสดงข้อมูลในรูปแบบต่าง ๆ ดังรูป



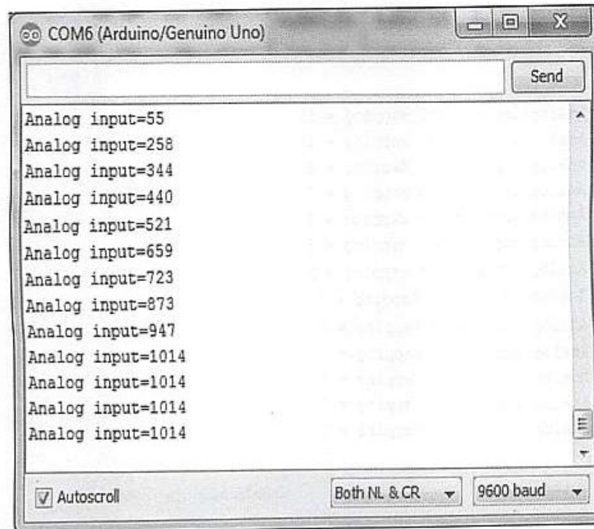
รูปที่ 7.5 ผลการรันโปรแกรมแสดงข้อมูลผ่าน Serial Monitor

ตัวอย่างที่ 7.2 โปรแกรมรับค่าสัญญาณแอนะล็อกจากความต้านทานปรับค่าได้มาแสดงผล

```
void setup()
{  Serial.begin(9600);
   pinMode(A0,INPUT);
}
void loop()
{
   Serial.print("Analog input=");
   Serial.println(analogRead(A0));
   delay(500);
}
```

ผลการรันโปรแกรม

กำหนดให้ขา A0 เป็นขาอินพุต จากนั้นอ่านค่าสัญญาณแอนะล็อกจากขาสัญญาณ A0 แล้วนำมาแสดงผล ซึ่งจะมีการอ่านค่าทุก ๆ 500 ms โดยค่าที่อ่านได้อยู่ในช่วง 0-1023 ตามการปรับค่าความต้านทาน



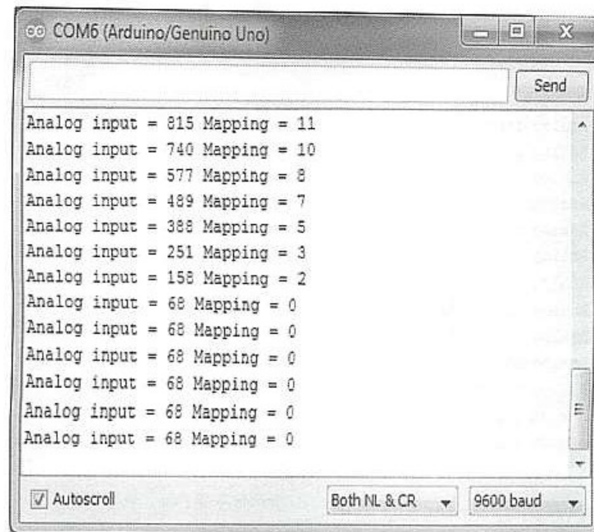
รูปที่ 7.6 การอ่านค่าสัญญาณแอนะล็อกจากความต้านทานปรับค่าได้

ตัวอย่างที่ 7.3 โปรแกรมรับค่าสัญญาณแอนะล็อกมาแสดงผลผ่านฟังก์ชัน map();

```
void setup()
{ Serial.begin(9600);
  pinMode(A0,INPUT);
}
void loop()
{  int VR=analogRead(A1);
   Serial.print(“Analog input = ”);
   Serial.print( VR );
   Serial.print(“Mapping = ”);
   Serial.println(map(VR,0,1023,0,15));
   delay(500);
}
```

ผลการรันโปรแกรม

กำหนดให้ขา A1 เป็นขาอินพุต ทำการอ่านค่าสัญญาณแอนะล็อกจากขาสัญญาณ A1 นำมาเก็บไว้ในตัวแปร VR จากนั้นแปลงค่าที่อ่านได้ในช่วง 0-1023 ให้อยู่ในช่วง 0-15 แล้วนำมาแสดงผล



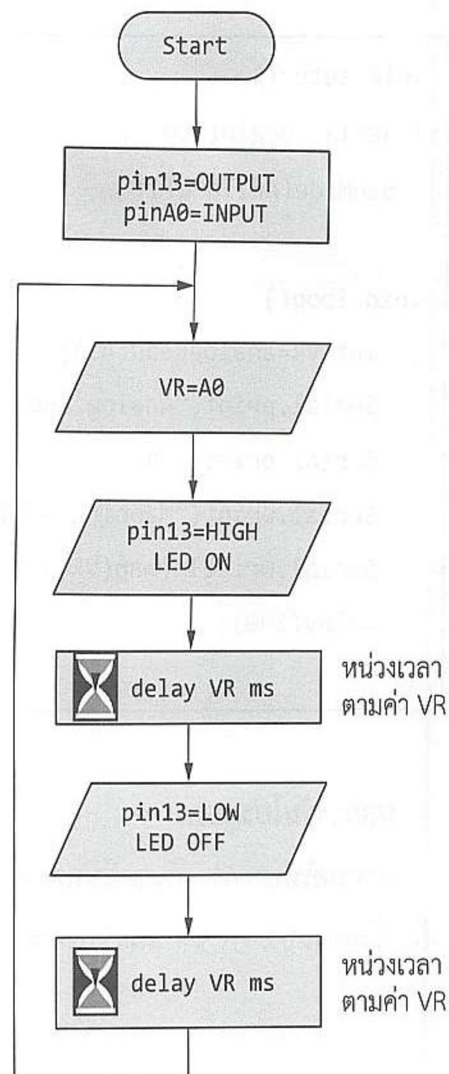
รูปที่ 7.7 การอ่านค่าสัญญาณแอนะล็อกผ่านฟังก์ชัน map();

ตัวอย่างที่ 7.4 โปรแกรมไฟกะพริบตามค่าของความต้านทาน

```
void setup()
{
  pinMode(13,OUTPUT);
  pinMode(A0,INPUT);
}
void loop()
{
   int VR=analogRead(A0);

   digitalWrite(13,HIGH);
   delay(VR);

   digitalWrite(13,LOW);
   delay(VR);
}
```



รูปที่ 7.8 โฟลว์ชาร์ตไฟกะพริบตามค่า VR

ผลการรันโปรแกรม

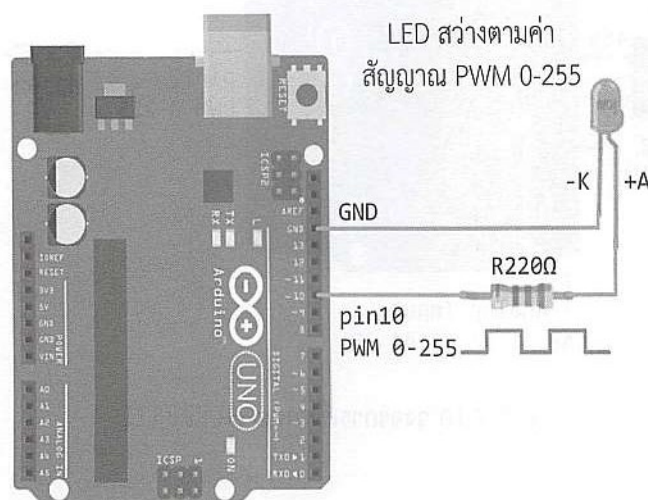
โปรแกรมจะวนรอบส่งสัญญาณ HIGH หรือลอจิก 1 และสัญญาณ LOW หรือลอจิก 0 ออกขา 13 ทำให้เกิดไฟกะพริบ โดยความเร็วของการกะพริบจะขึ้นอยู่กับการปรับค่าความต้านทาน (VR) ซึ่งมีค่าอยู่ในช่วง 0-1023

ตัวอย่างที่ 7.5 โปรแกรมปรับความสว่างของหลอดแสดงผล

```
void setup()
{ pinMode(10,OUTPUT);
}
void loop()
 { int P;
   for(P=0;P<=255;P=P+5)
    { analogWrite(10,P);
      delay(50);
    }
    analogWrite(10,0)
    delay(1000);
 }
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบส่งสัญญาณ PWM ออกขา 10 (analogWrite(10,P);) โดยค่าของสัญญาณ PWM ที่ส่งออกเริ่มจาก 0 แล้วเพิ่มขึ้นทีละ 5 จนถึง 255 ทำให้หลอดแสดงผล LED ค่อย ๆ สว่างขึ้น



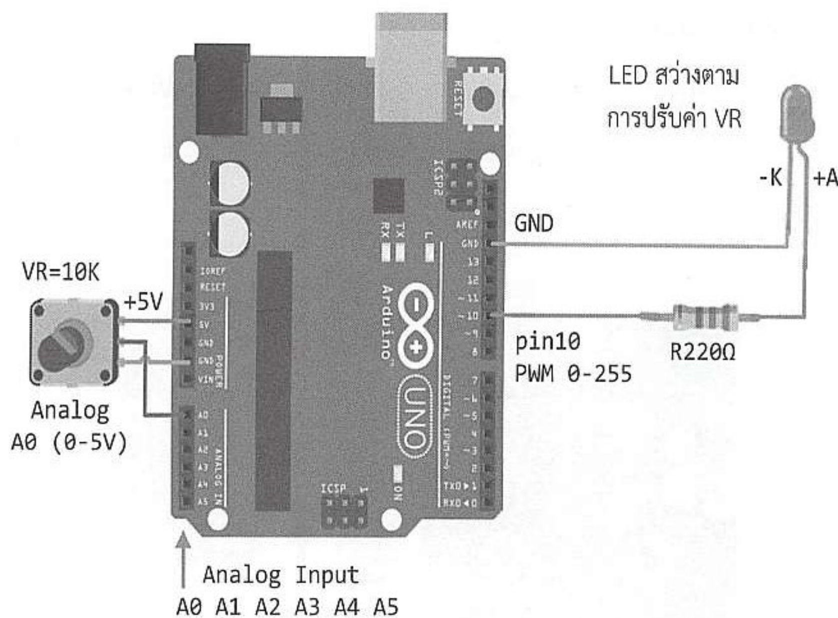
รูปที่ 7.9 การส่งสัญญาณ PWM ออกขา 10

ตัวอย่างที่ 7.6 โปรแกรมปรับความสว่างตามค่าของความต้านทานแบบปรับค่าได้

```
void setup()
 {
   pinMode(A0,INPUT);
   pinMode(10,OUTPUT);
 }
void loop()
 { int VR=analogRead(A0);
   VR=map(VR,0,1023,0,255);
   analogWrite(10,VR);
 }
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบส่งสัญญาณ PWM ออกขา 10 โดยค่าของสัญญาณ PWM ที่ส่งออกอยู่ในช่วง 0-255 ทำให้หลอดแสดงผล LED ที่ต่อกับขา 10 สว่างตามการปรับค่าความต้านทานแบบปรับค่าได้ (VR)

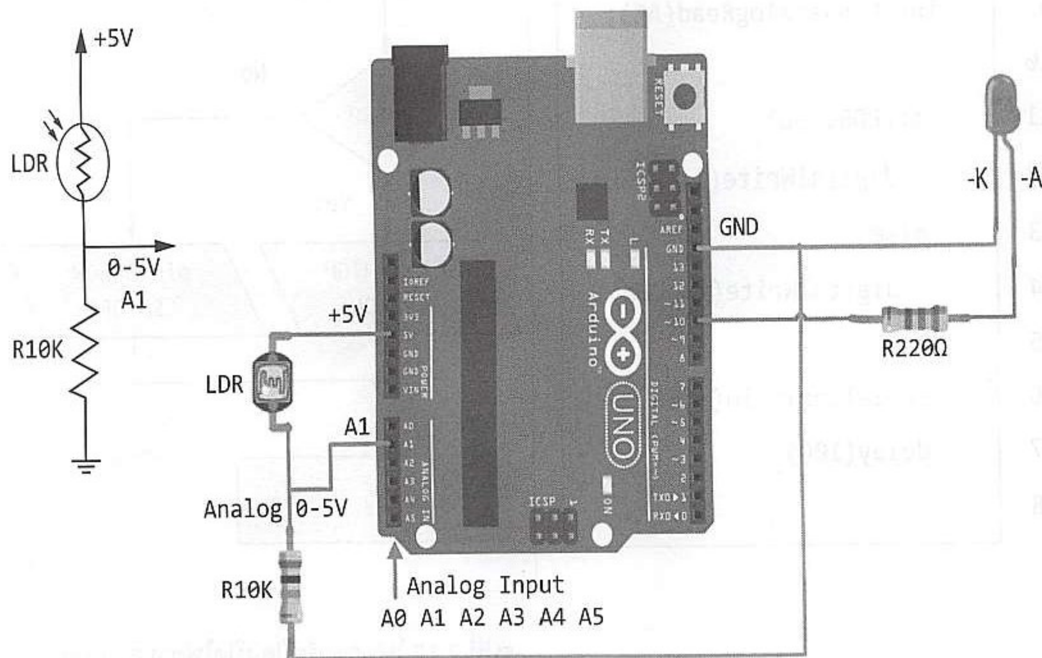


รูปที่ 7.10 วงจรการเชื่อมต่อ VR และ LED

7.5 ตัวต้านทานแบบ LDR

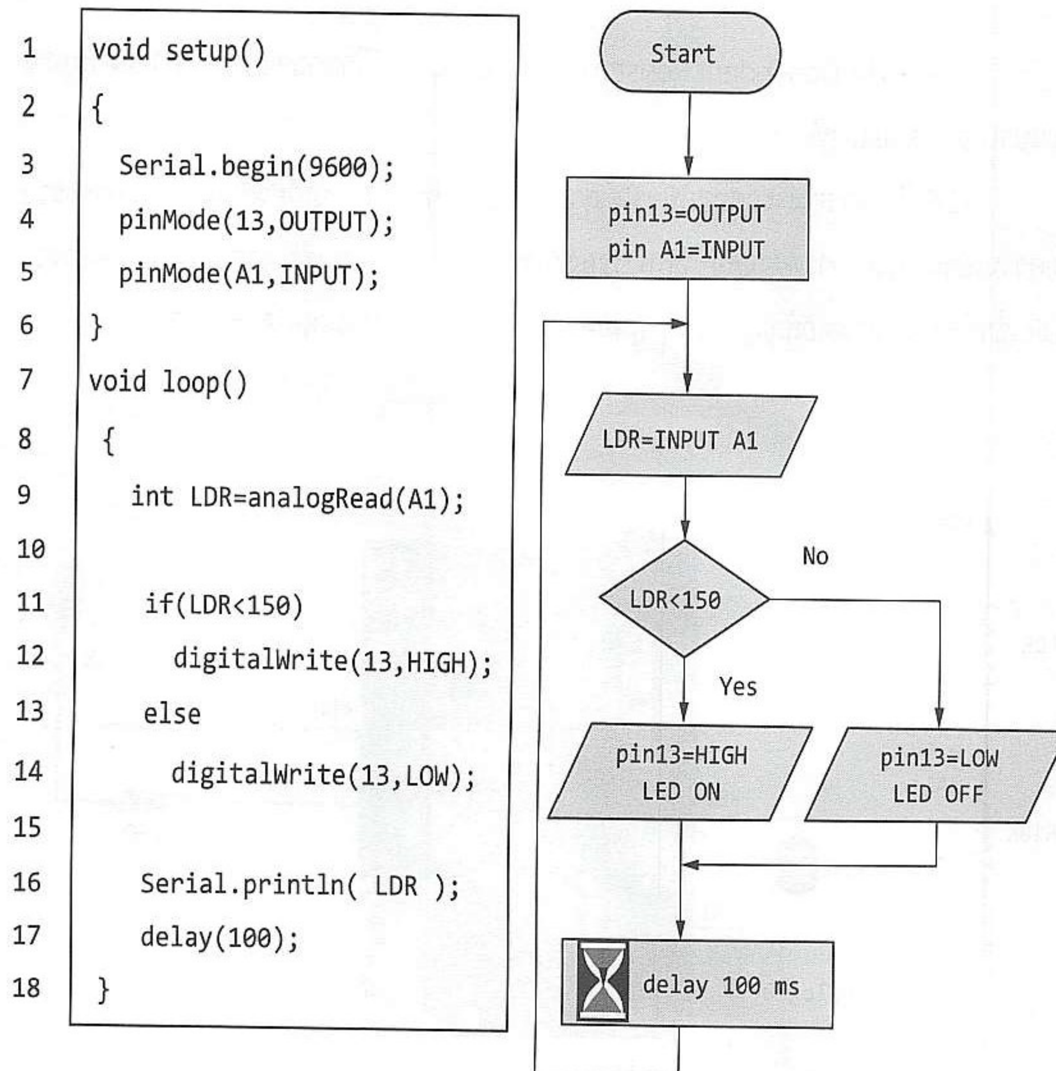
LDR (Light Dependent Resistor) คือความต้านทานที่ไวต่อแสง โดยค่าความต้านทานจะแปรเปลี่ยนตามแสงที่ตกกระทบ

LDR มี 2 ขา สามารถต่อสลับขาได้ ส่วนมากประยุกต์ใช้ในงานเกี่ยวกับแสง เช่น การตรวจวัดความเข้มของแสง การตรวจสอบทิศทางของแสงหรือกลางวัน-กลางคืน จากรูปเป็นการต่อวงจร LDR เข้าที่ขา A1 และต่อสัญญาณเอาต์พุตออกขา 10 เพื่อไปควบคุมหลอดแสดงผล LED



รูปที่ 7.11 วงจรการเชื่อมต่อ LDR กับบอร์ด Arduino

ตัวอย่างที่ 7.7 โปรแกรมปิด-เปิดไฟตามค่า LDR



รูปที่ 7.12 โฟลว์ชาร์ตปิด-เปิดไฟตามค่า LDR

ผลการรันโปรแกรม

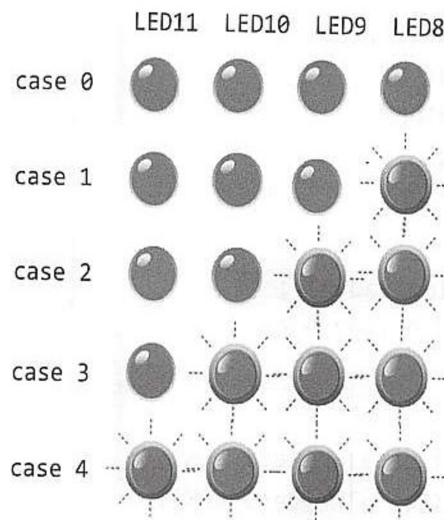
ถ้าไม่มีแสงหรือมืด ไฟจะติด และถ้าสว่าง ไฟจะดับ โดยโปรแกรมจะวนรอบอ่านข้อมูลจากขา A1 (LDR=analogRead(A1);) จากนั้นนำค่าของ LDR มาเปรียบเทียบ ถ้า LDR น้อยกว่า 150 (แสดงว่ามืด) ให้ส่งสัญญาณ HIGH ออกขา 13 แล้วทำให้ไฟติด ถ้าไม่ใช่ ให้ส่งสัญญาณ LOW ออกขา 13 และทำให้ไฟดับ

ตัวอย่างที่ 7.8 โปรแกรมรับค่าจาก LDR แสดงผลใน LED 4 ดวง

```
void setup()
 {
   Serial.begin(9600);
   pinMode(8,OUTPUT);
   pinMode(9,OUTPUT);
   pinMode(10,OUTPUT);
   pinMode(11,OUTPUT);
   pinMode(A1,INPUT);
 }
void loop()
 {  int LDR=analogRead(A1);
    LDR=(map(LDR,0,1023,0,4));
    switch(LDR)
      {
       case 0: digitalWrite(8,LOW);
               digitalWrite(9,LOW);
               digitalWrite(10,LOW);
               digitalWrite(11,LOW);
               break;
       case 1: digitalWrite(8,HIGH);
               digitalWrite(9,LOW);
               digitalWrite(10,LOW);
               digitalWrite(11,LOW);
               break;
       case 2: digitalWrite(8,HIGH);
               digitalWrite(9,HIGH);
               digitalWrite(10,LOW);
               digitalWrite(11,LOW);
               break;
       case 3: digitalWrite(8,HIGH);
               digitalWrite(9,HIGH);
               digitalWrite(10,HIGH);
               digitalWrite(11,LOW);
               break;
       case 4: digitalWrite(8,HIGH);
               digitalWrite(9,HIGH);
               digitalWrite(10,HIGH);
               digitalWrite(11,HIGH);
               break;
      }
      delay(100);
 }
```

ผลการรันโปรแกรม

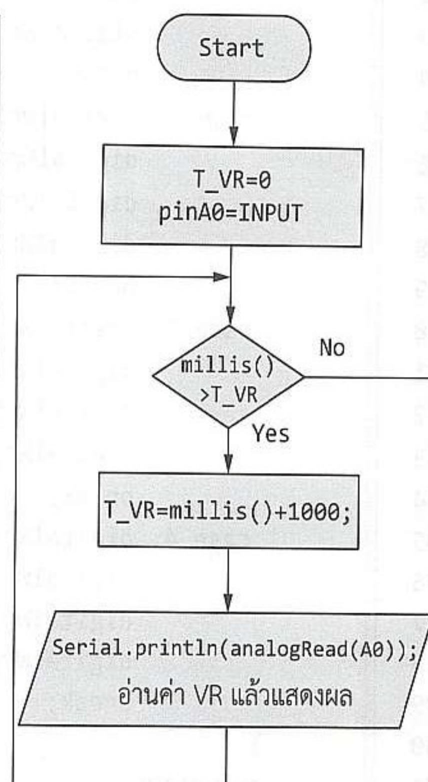
ค่าความต้านทาน LDR ที่อ่านได้คือ 0-1023 แล้วปรับให้อยู่ในช่วง 0-4 แล้วส่งค่าให้กับฟังก์ชัน switch() มี 5 กรณี คือ 0-4 ถ้าเป็น 0 หลอดไม่ติด ถ้าเป็น 1 หลอดติด 1 ดวง ... ถ้าเป็น 4 หลอดติด 4 ดวงตามลำดับ



รูปที่ 7.13 การทำงานของ case 0-4

ตัวอย่างที่ 7.9 โปรแกรมอ่านค่าตัวต้านทาน VR ทุก ๆ 1 วินาทีโดยใช้ฟังก์ชัน millis();

```
unsigned long T_VR;
void setup()
{
 Serial.begin(9600);
 pinMode(A0,INPUT);
}
void loop()
{
  if(millis()>T_VR)
  { T_VR=millis()+1000;
    Serial.println(analogRead(A0));
  }
}
```



รูปที่ 7.14 โฟลว์ชาร์ตอ่านค่า VR ทุก ๆ 1 วินาที

ผลการรันโปรแกรม

โปรแกรมจะวนรอบอ่านค่าสัญญาณแอนะล็อกที่ขา A0 ทุก ๆ 1000 ms หรือ 1 วินาที โดยใช้ฟังก์ชัน millis(); ทำให้ไมโครคอนโทรลเลอร์สามารถทำงานอื่นได้โดยไม่ติดอยู่ในฟังก์ชัน delay();

ตัวอย่างที่ 7.10 โปรแกรมอ่านค่า VR ทุก ๆ 1 วินาทีและ LDR ทุก ๆ 2 วินาทีโดยใช้ฟังก์ชัน millis();

```
unsigned long T_LDR,T_VR;
void setup()
{
 Serial.begin(9600);
 pinMode(A0,INPUT);
 pinMode(A1,INPUT);
}
void loop()
{
 if(millis()>T_VR)
 { T_VR=millis()+1000;
   Serial.println(analogRead(A0));
 }
if(millis()>T_LDR)
 { T_LDR=millis()+2000;
   Serial.println(analogRead(A1));
 }
}
```

ผลการรันโปรแกรม

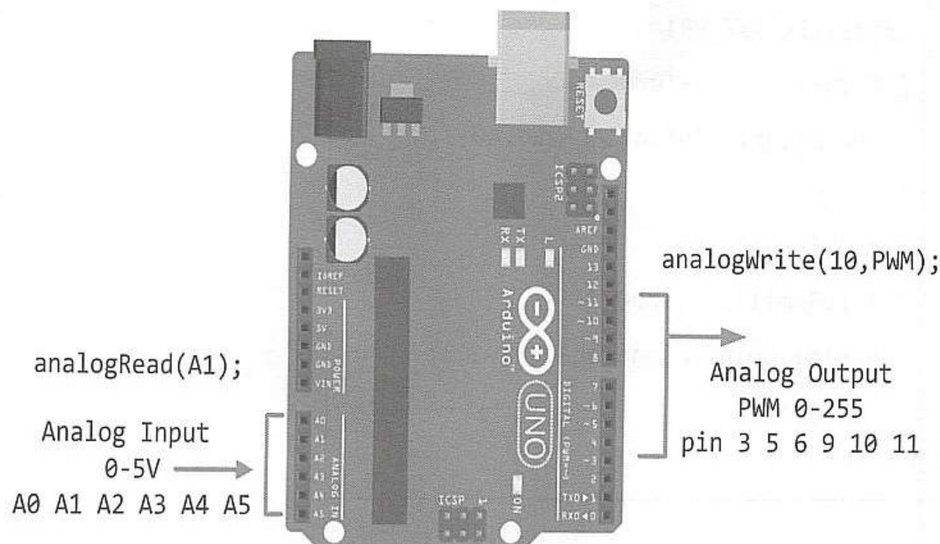
โปรแกรมจะวนรอบอ่านค่าสัญญาณแอนะล็อกที่ขา A0 ทุก ๆ 1000 ms หรือ 1 วินาที และอ่านค่าสัญญาณแอนะล็อกจากขา A1 ทุก ๆ 2000 ms หรือ 2 วินาทีโดยใช้ฟังก์ชัน millis(); ทำให้ไมโครคอนโทรลเลอร์สามารถทำงานอื่นได้โดยไม่ติดอยู่ในฟังก์ชัน delay();

7.6 สรุป

แอนะล็อก คือสัญญาณที่มีความต่อเนื่อง มีหลายค่า เป็นสัญญาณธรรมชาติต่างจากสัญญาณดิจิทัลที่มีเพียง 2 ระดับ สัญญาณแอนะล็อกมีลักษณะเป็นคลื่นที่มีความถี่และความเข้มของสัญญาณที่ต่างกันและมักเปลี่ยนแปลงตลอดเวลา มีทั้งสัญญาณที่มีการเปลี่ยนแปลงเป็นรูปแบบที่แน่นอน เช่น สัญญาณไซน์หรือสัญญาณไฟฟ้ากระแสสลับ และยังมีสัญญาณแอนะล็อกที่เกิดขึ้นโดยธรรมชาติ ซึ่งส่วนใหญ่จะมีลักษณะที่แตกต่างกัน เช่น สัญญาณเสียง แสง ความชื้น อุณหภูมิ

analogRead(); คือฟังก์ชันอ่านค่าข้อมูลแอนะล็อกจากขา A0-A5 ของไมโครคอนโทรลเลอร์ Arduino UNO ซึ่งมีค่าอยู่ระหว่าง 0-1023 โดยผ่านวงจรแปลงสัญญาณแอนะล็อกเป็นดิจิทัลขนาด 10 บิต (A/D 10 bits)

analogWrite(); คือฟังก์ชันส่งข้อมูลแอนะล็อกออกขาเอาต์พุต ซึ่งเป็นการส่งแบบ PWM ค่าที่ส่งออกอยู่ระหว่าง 0-255 ในบอร์ด Arduino UNO จะมีขาเอาต์พุต PWM อยู่ 6 ขา คือ ขา 3, 5, 6, 9, 10 และ 11 ส่วนสัญญาณ PWM คือสัญญาณที่มีการปรับความกว้างของสัญญาณพัลส์ตามสัดส่วนที่กำหนด



รูปที่ 7.15 แอนะล็อกอินพุตและแอนะล็อกเอาต์พุต

คำถามท้ายบทที่ 7

1. สัญญาณแอนะล็อกคืออะไร
2. สัญญาณแอนะล็อกต่างจากสัญญาณดิจิทัลอย่างไร
3. จงอธิบายฟังก์ชัน analogRead();
4. จงอธิบายฟังก์ชัน analogWrite();
5. จงอธิบายฟังก์ชัน map();
6. จงเขียนโปรแกรมอ่านค่า VR ทุก ๆ 1 วินาที อ่านค่า LDR ทุก ๆ 2 วินาที และอ่านค่าอุณหภูมิ ทุก ๆ 3 วินาที
7. จงเขียนโปรแกรมอ่านค่าจาก VR แล้วแบ่งเป็น 4 ช่วงดังนี้

   0-255 ให้เอาต์พุตขา 2 ทำงาน

   256-511 ให้เอาต์พุตขา 3 ทำงาน

   512-767 ให้เอาต์พุตขา 4 ทำงาน

   768-1023 ให้เอาต์พุตขา 5 ทำงาน

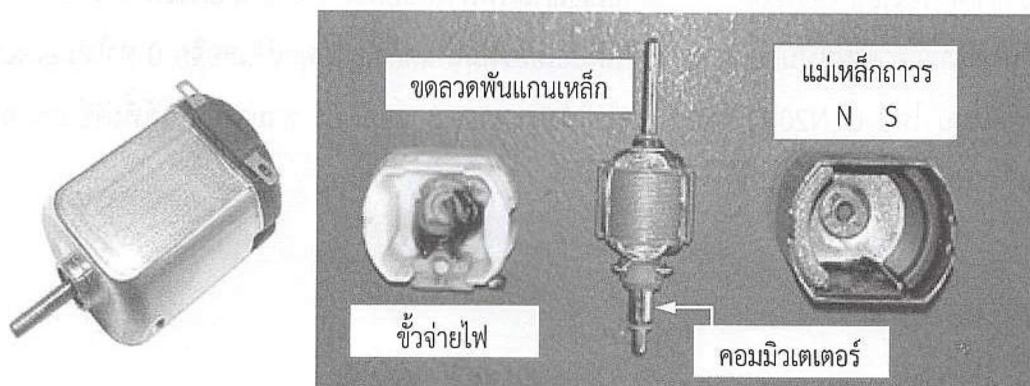
8

การควบคุมมอเตอร์

มอเตอร์ (Motor) คือเครื่องกลไฟฟ้าที่ทำหน้าที่เปลี่ยนพลังงานไฟฟ้าเป็นพลังงานกล การเคลื่อนที่ของมอเตอร์จะเคลื่อนที่ด้วยการหมุนซ้ายหรือหมุนขวา เพื่อขับเคลื่อนอุปกรณ์หรือระบบที่ต้องการควบคุม ในบทนี้จะกล่าวถึงการเขียนโปรแกรมควบคุมมอเตอร์ 3 ชนิด คือ ดีซีมอเตอร์ (DC Motor) สเต็ปเปอร์มอเตอร์ (Stepper Motor) และเซอร์โวมอเตอร์ (Servo Motor) เพื่อเป็นแนวทางในการประยุกต์ใช้งานไมโครคอนโทรลเลอร์ในเครื่องจักรกลต่าง ๆ

8.1 ดีซีมอเตอร์

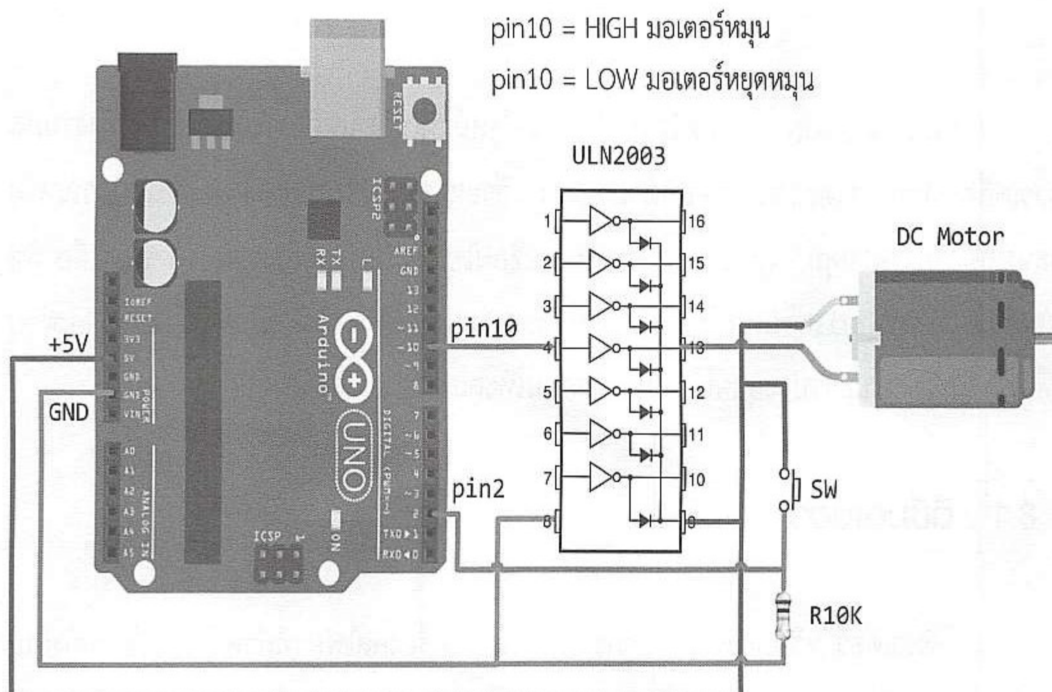
ดีซีมอเตอร์ หรือมอเตอร์ไฟฟ้ากระแสตรงเป็นเครื่องกลไฟฟ้าที่ทำหน้าที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานกล เคลื่อนที่ด้วยการหมุนโดยอาศัยการผลักกันของสนามแม่เหล็กไฟฟ้าและแม่เหล็กถาวร มีโครงสร้างหลักที่สำคัญคือ แม่เหล็กถาวร 2 อันจะติดรอบตัวถังของมอเตอร์ แกนเหล็กพันด้วยขดลวดทองแดงทำให้เกิดสนามแม่เหล็กไฟฟ้าเมื่อป้อนไฟ และคอมมิวเตเตอร์สำหรับรับสัญญาณไฟจากขั้วจ่ายไฟหรือแปรงถ่านซึ่งติดอยู่กับฝาปิดท้ายของดีซีมอเตอร์



รูปที่ 8.1 มอเตอร์ไฟฟ้ากระแสตรงขนาดเล็ก

หลักการทำงานของดีซีมอเตอร์ คือ เมื่อป้อนกระแสไฟเข้าขดลวดจะทำให้เกิดสนามแม่เหล็กไฟฟ้า และเกิดแรงผลักระหว่างแม่เหล็กถาวรกับแม่เหล็กไฟฟ้าทำให้มอเตอร์หมุน โดยความเร็วการหมุนของมอเตอร์จะขึ้นอยู่กับแรงดันไฟฟ้าที่ป้อนให้มอเตอร์ ขนาดของขดลวด จำนวนรอบพันขดลวด และความเข้มของเส้นแรงแม่เหล็กถาวร

8.2 วงจรการเชื่อมต่อดีซีมอเตอร์กับ Arduino

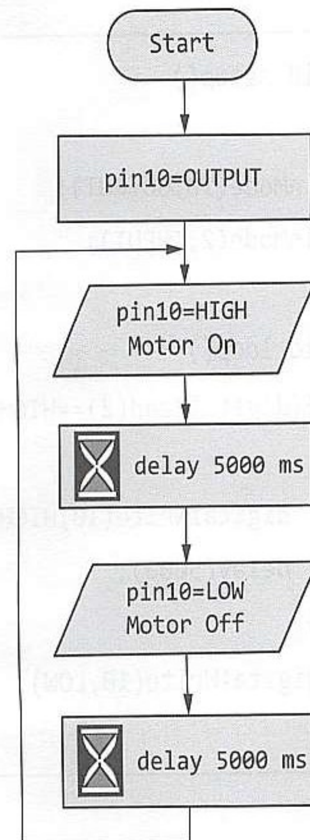


รูปที่ 8.2 วงจรการเชื่อมต่อดีซีมอเตอร์กับ Arduino

จากรูป เชื่อมต่อเอาต์พุตขา 10 ของไมโครคอนโทรลเลอร์ซึ่งเป็นขาสัญญาณ PWM เข้าขา 4 ของไอซี ULN2003 เพื่อขยายสัญญาณไฟฟ้าไปขับดีซีมอเตอร์ โดยการทำงานของวงจร ถ้าสัญญาณเอาต์พุตเป็นลอจิก 1 จะทำให้มอเตอร์หมุน แต่ถ้าเอาต์พุตเป็นลอจิก 0 ทำให้มอเตอร์หยุดหมุน ไอซี ULN2003 ทนกระแสไฟได้ประมาณ 3 แอมแปร์ สามารถขับได้ทั้งดีซีมอเตอร์ สเต็ปเปอร์มอเตอร์ และรีเลย์

ตัวอย่างที่ 8.1 โปรแกรมควบคุมให้มอเตอร์หมุนและหยุดหมุนสลับกัน

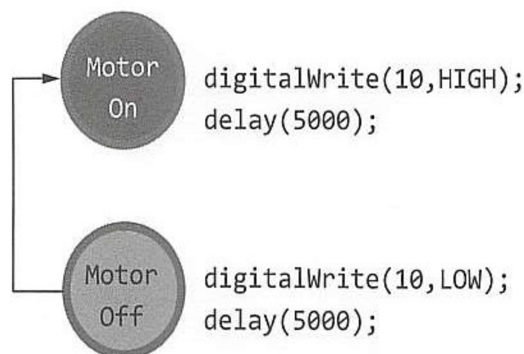
```
void setup()
{
 pinMode(10,OUTPUT);
}
void loop()
{
   digitalWrite(10,HIGH);
   delay(5000);
   digitalWrite(10,LOW);
   delay(5000);
 }
```



รูปที่ 8.3 โฟลว์ชาร์ตควบคุมการปิด-เปิดมอเตอร์

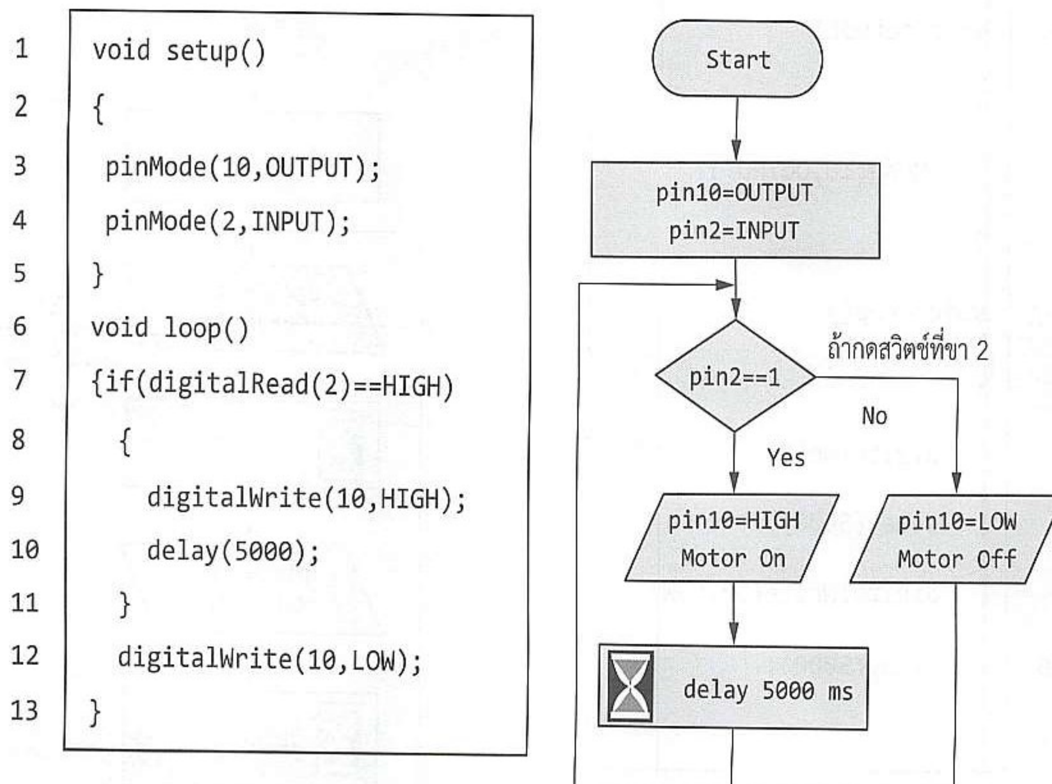
ผลการรันโปรแกรม

โปรแกรมส่งสัญญาณให้ขา 10 เป็น HIGH หรือลอจิก 1 ทำให้มอเตอร์หมุนเป็นเวลา 5000 ms หรือ 5 วินาที จากนั้นส่งสัญญาณ LOW หรือลอจิก 0 ทำให้มอเตอร์หยุดหมุนเป็นเวลา 5 วินาทีสลับกันไปมา



รูปที่ 8.4 มอเตอร์จะหมุนและหยุดหมุนสลับกันอย่างละ 5 วินาที

ตัวอย่างที่ 8.2 โปรแกรมควบคุมให้มอเตอร์ทำงาน 5 วินาทีตามการกดสวิตช์



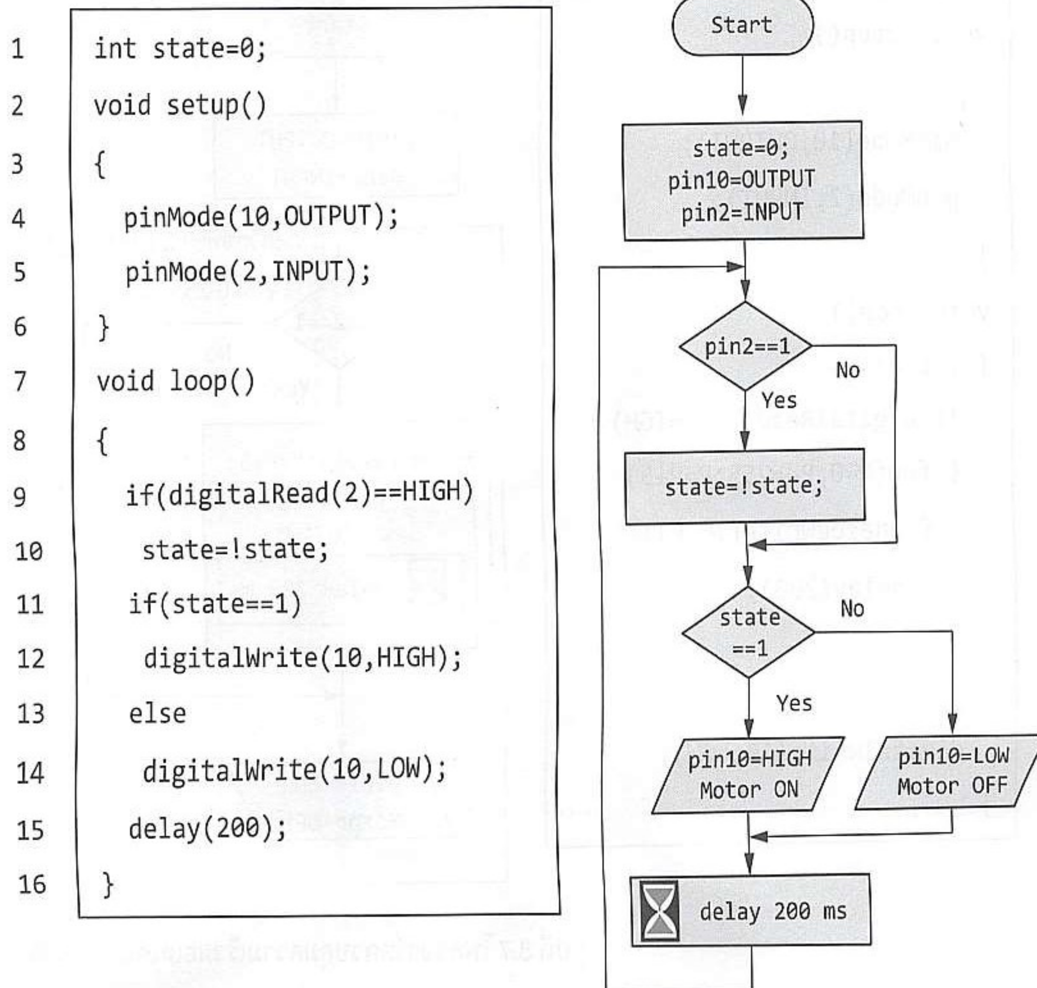
```
void setup()
{
 pinMode(10,OUTPUT);
 pinMode(2,INPUT);
}
void loop()
{if(digitalRead(2)==HIGH)
  {
    digitalWrite(10,HIGH);
    delay(5000);
  }
  digitalWrite(10,LOW);
}
```


รูปที่ 8.5 โฟลว์ชาร์ตควบคุมมอเตอร์ให้ทำงาน 5 วินาทีตามการกดสวิตช์

ผลการรันโปรแกรม

เมื่อกดสวิตช์ที่ขา 2 จะทำให้ขา 10 เป็น HIGH หรือลอจิก 1 ทำให้มอเตอร์หมุนเป็นเวลา 5000 ms หรือ 5 วินาที

ตัวอย่างที่ 8.3 โปรแกรมเปิด-ปิดมอเตอร์ตามการกดสวิตช์



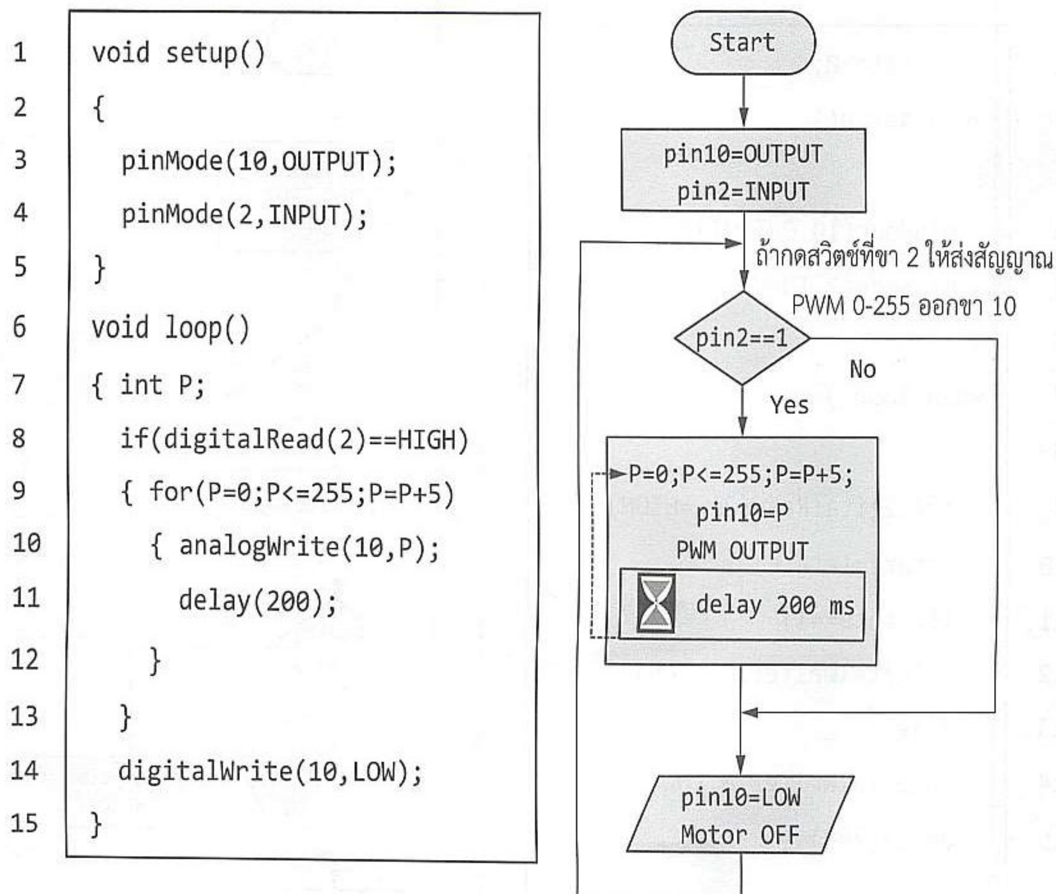
```
int state=0;
void setup()
{
  pinMode(10,OUTPUT);
  pinMode(2,INPUT);
}
void loop()
{
  if(digitalRead(2)==HIGH)
   state=!state;
  if(state==1)
   digitalWrite(10,HIGH);
  else
   digitalWrite(10,LOW);
  delay(200);
}
```


รูปที่ 8.6 โฟลว์ชาร์ตการเปิด-ปิดมอเตอร์ตามการกดสวิตช์

ผลการรันโปรแกรม

เมื่อกดสวิตช์ที่ขา 2 จะกลับสถานะของตัวแปร state โดยถ้าตัวแปร state เท่ากับ 1 ให้มอเตอร์หมุน แต่ถ้าตัวแปร state เท่ากับ 0 ให้มอเตอร์หยุดหมุน ดังนั้นทุกครั้งที่มีการกดสวิตช์การทำงานของมอเตอร์จะเปลี่ยนสถานะจากปิดเป็นเปิด หรือเปิดเป็นปิด

ตัวอย่างที่ 8.4 โปรแกรมควบคุมความเร็วมอเตอร์แบบ PWM



```
void setup()
{
  pinMode(10,OUTPUT);
  pinMode(2,INPUT);
}
void loop()
{ int P;
  if(digitalRead(2)==HIGH)
  { for(P=0;P<=255;P=P+5)
    { analogWrite(10,P);
      delay(200);
    }
  }
  digitalWrite(10,LOW);
}
```


รูปที่ 8.7 โฟลว์ชาร์ตควบคุมความเร็วมอเตอร์แบบ PWM

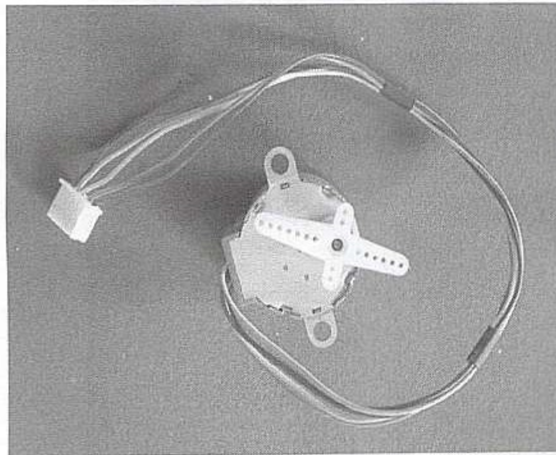
ผลการรันโปรแกรม

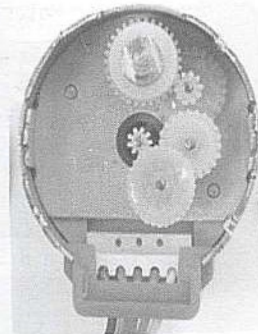
เมื่อกดสวิตช์ที่ขา 2 จะทำให้โปรแกรมวนรอบส่งข้อมูล 0-255 โดยเริ่มจาก 0 และเพิ่มขึ้นทีละ 5 จนถึง 255 ออกทางขา 10 ซึ่งเป็นการส่งสัญญาณแบบ PWM ทำให้มอเตอร์ค่อย ๆ หมุนจนถึงความเร็วสูงสุด (PWM = 255) แล้วหยุดการทำงาน

8.3 สเต็ปเปอร์มอเตอร์

สเต็ปเปอร์มอเตอร์ เป็นมอเตอร์ที่มีลักษณะการหมุนเป็นขั้น ๆ ไม่ต่อเนื่อง สามารถควบคุมตำแหน่งหรือมุมในการหมุนได้ โดยปกติจะหมุนขั้นละ 1.8, 5 หรือ 7.5 องศา ซึ่งสามารถดูรายละเอียดได้จากแผ่นป้ายที่ติดกับสเต็ปเปอร์มอเตอร์ การควบคุมการหมุนของสเต็ปเปอร์มอเตอร์สามารถทำได้โดยการป้อนสัญญาณพัลส์ให้กับสเต็ปเปอร์มอเตอร์ ซึ่งจะทำงานแบบลำดับตามสัญญาณพัลส์ที่ป้อนเข้า

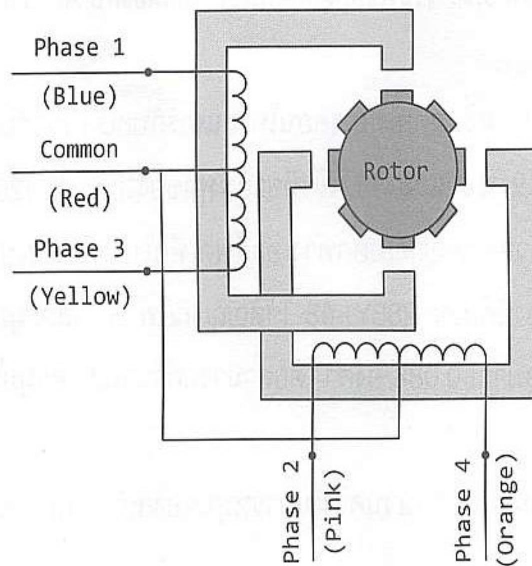
ในการทดลองนี้ใช้สเต็ปเปอร์ที่หมุนขั้นละ 5.625 องศา ส่วนโครงสร้างภายในมีเฟืองทดอัตราส่วน 64:1 ทำให้การหมุน 1 รอบหรือ 360 องศาต้องส่งสัญญาณพัลส์ 4096 ครั้ง





เฟืองทด 64:1
360 องศา = 4096 พัลส์

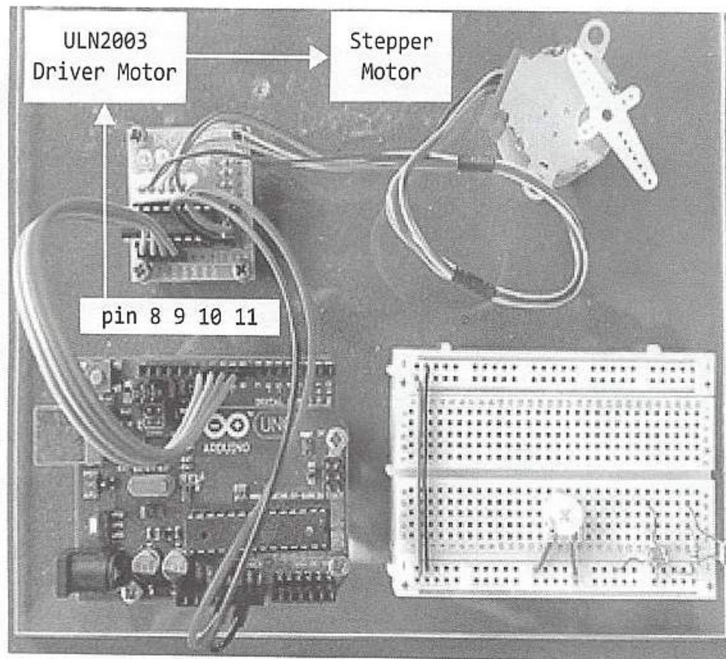
รูปที่ 8.8 สเต็ปเปอร์มอเตอร์



รูปที่ 8.9 โครงสร้างของสเต็ปเปอร์มอเตอร์

สเต็ปเปอร์มอเตอร์ประกอบไปด้วยองค์ประกอบสำคัญ 2 ส่วน คือ โรเตอร์ (Rotor) เป็นส่วนที่หมุนได้ เป็นแม่เหล็กถาวร มีลักษณะเป็นทรงกระบอกแบบเซาะร่อง และอีกส่วนคือ สเตเตอร์ (Stator) เป็นส่วนที่ติดกับตัวถังมีขดลวดต่ออนุกรมอยู่ 4 ขดหรือ 4 เฟส ซึ่งต้องป้อนสัญญาณพัลส์เข้าเฟส 1 ถึงเฟส 4 และขาร่วม (Common) อีก 1 หรือ 2 สายต้องต่อกับสัญญาณไฟ

8.4 วงจรการเชื่อมต่อสเต็ปเปอร์มอเตอร์กับ Arduino



รูปที่ 8.10 วงจรเชื่อมต่อสเต็ปเปอร์มอเตอร์กับ Arduino

จากรูปที่ 8.10 การเชื่อมต่อไมโครคอนโทรลเลอร์กับสเต็ปเปอร์มอเตอร์ สัญญาณควบคุมจะถูกส่งออกจากขา 8, 9, 10 และ 11 เข้าที่ขาอินพุตของไอซี ULN2003 เพื่อทำหน้าที่ขยายแรงดันและกระแสไฟแล้วส่งสัญญาณออกทางเอาต์พุตเพื่อไปขับสเต็ปเปอร์มอเตอร์ สเต็ปเปอร์มอเตอร์ที่ใช้ในการทดลองนี้คือรุ่น 28BYJ-48 ใช้สัญญาณไฟ 5 โวลต์ มุมต่อการหมุน 1 ครั้งคือ 5.625/64 องศาหรือประมาณ 0.088 องศา โดยสามารถคำนวณหาค่ามุมในการหมุนดังนี้

ตัวอย่าง จงคำนวณหาจำนวนครั้งในการหมุนของสเต็ปเปอร์มอเตอร์ในการหมุน 1 รอบหรือ 360 องศา

จำนวนครั้ง = 360/(5.625/64)

= 4096

หมายถึง ต้องจ่ายสัญญาณพัลส์จำนวน 4096 สเต็ปหรือครั้งจึงทำให้สเต็ปเปอร์มอเตอร์หมุนได้ 1 รอบหรือ 360 องศา หรือการหมุน 1 องศาต้องป้อนสัญญาณพัลส์ 11.38 ครั้ง (4096/360)

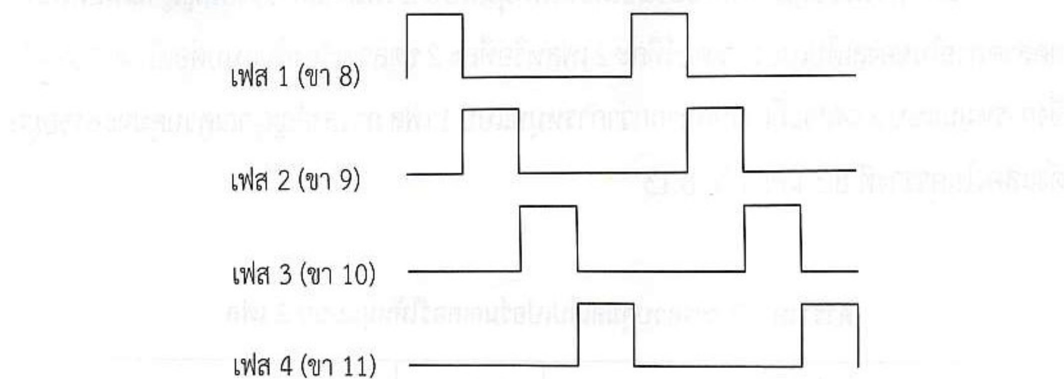
8.5 การควบคุมสเต็ปเปอร์มอเตอร์

การควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนต้องส่งสัญญาณพัลส์หรือสัญญาณไฟฟ้าแบบลำดับให้กับขดลวดทั้ง 4 ขดของสเต็ปเปอร์มอเตอร์ ซึ่งมีการควบคุมแบบ 1 เฟส 2 เฟส และแบบครึ่งขั้น ดังนี้

1) **การควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนแบบ 1 เฟส** คือการจ่ายสัญญาณพัลส์ให้กับขดลวดของสเต็ปเปอร์มอเตอร์ทีละเฟสหรือทีละขดลวดเรียงกันแบบต่อเนื่อง โดยการส่งข้อมูลลอจิก 1 ให้ขา 8, 9, 10 และ 11 ตามลำดับ แสดงดังตารางที่ 8.1 และรูปที่ 8.11

ตารางที่ 8.1 การควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนแบบ 1 เฟส

ลำดับ	1	2	3	4
เฟส 1 (ขา 8)	1	0	0	0
เฟส 2 (ขา 9)	0	1	0	0
เฟส 3 (ขา 10)	0	0	1	0
เฟส 4 (ขา 11)	0	0	0	1

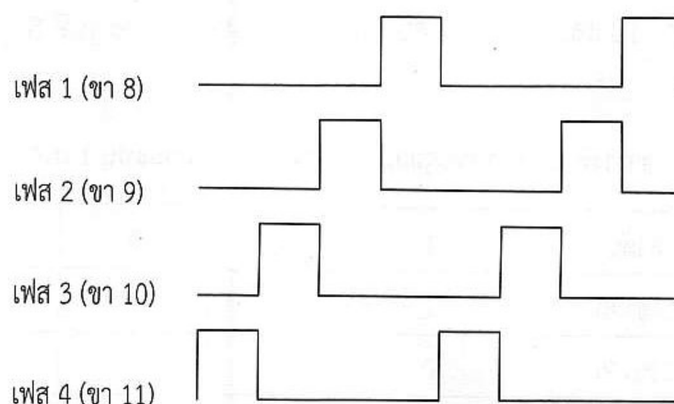


รูปที่ 8.11 สัญญาณควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนแบบ 1 เฟส

การกลับทิศทางหมุนของสเต็ปเปอร์มอเตอร์สามารถทำได้โดยควบคุมการจ่ายสัญญาณพัลส์หรือข้อมูลแบบลำดับจากขา 8, 9, 10 และ 11 ให้กลับลำดับของขาสัญญาณเป็น 11, 10, 9 และ 8 ตามลำดับ ก็จะทำให้สเต็ปเปอร์มอเตอร์หมุนกลับทาง แสดงดังตารางที่ 8.2 และรูปที่ 8.12

ตารางที่ 8.2 การควบคุมการกลับทิศทางหมุนของสเต็ปเปอร์มอเตอร์แบบ 1 เฟส

ลำดับ	1	2	3	4
เฟส 1 (ขา 8)	0	0	0	1
เฟส 2 (ขา 9)	0	0	1	0
เฟส 3 (ขา 10)	0	1	0	0
เฟส 4 (ขา 11)	1	0	0	0

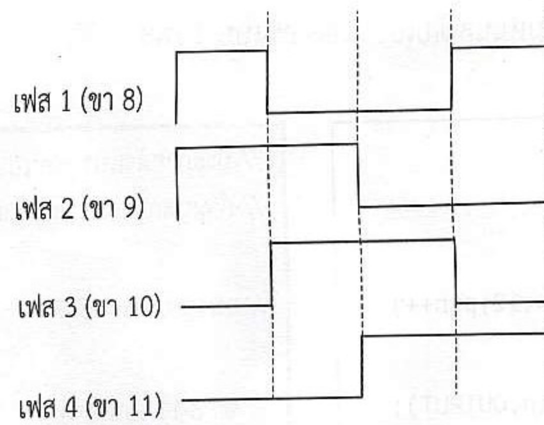


รูปที่ 8.12 สัญญาณควบคุมการกลับทิศทางหมุนของสเต็ปเปอร์มอเตอร์แบบ 1 เฟส

2) การควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนแบบ 2 เฟส คือการจ่ายสัญญาณพัลส์ให้กับขดลวดภายในของสเต็ปเปอร์มอเตอร์ทีละ 2 เฟสหรือทีละ 2 ขดลวดเรียงกันแบบต่อเนื่องตามลำดับ ซึ่งการหมุนแบบ 2 เฟสจะมีแรงบิดมากกว่าการหมุนแบบ 1 เฟส การส่งสัญญาณควบคุมจะส่งข้อมูลดังแสดงในตารางที่ 8.3 และรูปที่ 8.13

ตารางที่ 8.3 การควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนแบบ 2 เฟส

ลำดับ	1	2	3	4
เฟส 1 (ขา 8)	1	0	0	1
เฟส 2 (ขา 9)	1	1	0	0
เฟส 3 (ขา 10)	0	1	1	0
เฟส 4 (ขา 11)	0	0	1	1

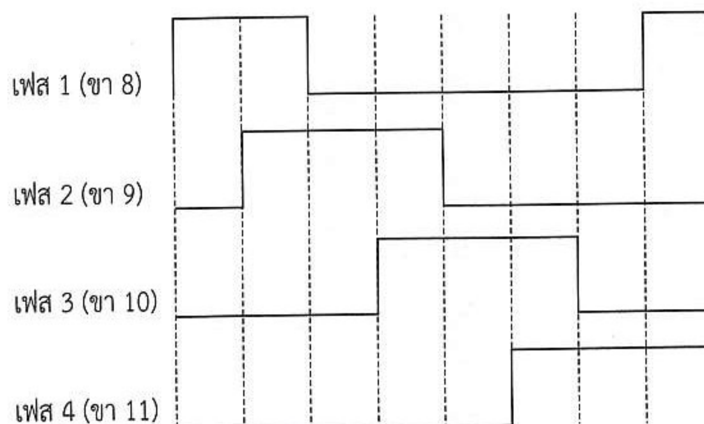


รูปที่ 8.13 สัญญาณควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนแบบ 2 เฟส

3) **การควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนแบบครึ่งขั้น** คือการควบคุมแบบผสมกันระหว่างการหมุนแบบ 1 เฟสและ 2 เฟสสลับกันไปมาตามลำดับ 1 ถึง 8 ซึ่งการหมุนแบบครึ่งขั้นจะให้มุมและองศาในการหมุนที่ละเอียดกว่าสองเท่า โดยการส่งข้อมูลแสดงในตารางที่ 8.4 และรูปที่ 8.14

ตารางที่ 8.4 สัญญาณควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนแบบครึ่งขั้น

ลำดับ	1	2	3	4	5	6	7	8
เฟส 1 (ขา 8)	1	1	0	0	0	0	0	1
เฟส 2 (ขา 9)	0	1	1	1	0	0	0	0
เฟส 3 (ขา 10)	0	0	0	1	1	1	0	0
เฟส 4 (ขา 11)	0	0	0	0	0	1	1	1



รูปที่ 8.14 สัญญาณควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนแบบครึ่งขั้น

ตัวอย่างที่ 8.5 โปรแกรมหมุนสเต็ปเปอร์มอเตอร์แบบ 1 เฟส

```
int pin;                          // ประกาศตัวแปร pin เป็นชนิดจำนวนเต็ม
void setup()                      // กำหนดค่าและขาสัญญาณ
{
 for(pin=8;pin<12;pin++)          // วนรอบ pin = 8-11
  {
    pinMode(pin,OUTPUT);          // ขา 8-11 เป็นเอาต์พุต
    digitalWrite(pin,LOW);        // ขา 8-11 เป็น LOW หรือ 0
  }
}
void loop()                       // วนรอบตลอดกาล
{
 for(pin=8;pin<12;pin++)          // วนรอบ pin = 8-11
  {
    digitalWrite(pin,HIGH);       // pin เป็น HIGH หรือ 1
    delay(5);                     // หน่วงเวลา 5 ms
    digitalWrite(pin,LOW);        // pin เป็น LOW หรือ 0
  }
}
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบส่งข้อมูล HIGH หรือลอจิก 1 ออกทางขา 8, 9, 10 และ 11 ทีละขาเลื่อนไปเรื่อย ๆ เหมือนโปรแกรมไฟวิ่ง 1 ดวง ทำให้สเต็ปเปอร์มอเตอร์หมุนแบบ 1 เฟส

ตัวอย่างที่ 8.6 โปรแกรมควบคุมสเต็ปเปอร์มอเตอร์แบบ 2 เฟส

```
void setup()
{ for(int pin=8;pin<12;pin++)
  { pinMode(pin,OUTPUT);
    digitalWrite(pin,LOW);
  }
  digitalWrite(8,HIGH);
  digitalWrite(9,HIGH);
}
void loop()
{ digitalWrite(8,LOW);
  digitalWrite(10,HIGH);
  delay(5);
  digitalWrite(9,LOW);
  digitalWrite(11,HIGH);
  delay(5);
  digitalWrite(10,LOW);
  digitalWrite(8,HIGH);
  delay(5);
  digitalWrite(11,LOW);
  digitalWrite(9,HIGH);
  delay(5);
}
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบส่งข้อมูล HIGH หรือลอจิก 1 ออกทางขา 8, 9, 10 และ 11 ทีละคู่เลื่อนไปเรื่อย ๆ เหมือนโปรแกรมไฟวิ่งทีละ 2 ดวง ทำให้สเต็ปเปอร์มอเตอร์หมุนแบบ 2 เฟส

ตัวอย่างที่ 8.7 โปรแกรมควบคุมความเร็วของสเต็ปเปอร์มอเตอร์ตาม VR

```
int pin;
void setup()                        // กำหนดค่าและขาสัญญาณ
{
  for(pin=8;pin<12;pin++)           // วนรอบ pin = 8-11
   { pinMode(pin,OUTPUT);           // ให้ขา 8-11 เป็นเอาต์พุต
     digitalWrite(pin,LOW);         // ให้ขา 8-11 เป็น LOW หรือ 0
   }
  pinMode(A0,INPUT);                // ให้ขา A0 เป็นอินพุต
}
void loop()                         // วนรอบตลอดกาล
{ int VR=analogRead(A0);            // อ่านข้อมูลจากขา A0 เก็บไว้ในตัวแปร VR
   VR=map(VR,0,1023,5,50);          // แปลงค่า VR จากช่วง 0-1023 เป็น 5-50
   for(pin=8;pin<12;pin++)          // วนรอบ pin = 8-11
   {
    digitalWrite(pin,HIGH);         // pin เป็น HIGH หรือ 1
    delay(VR);                      // หน่วงเวลาตามค่า VR
    digitalWrite(pin,LOW);          // pin เป็น LOW หรือ 0
   }
}
```

ผลการรันโปรแกรม

โปรแกรมจะวนรอบส่งข้อมูล HIGH หรือลอจิก 1 ออกทางขา 8, 9, 10 และ 11 ทีละขาเลื่อนไปเรื่อย ๆ เหมือนไฟวิ่ง 1 ดวง ทำให้สเต็ปเปอร์มอเตอร์หมุนแบบ 1 เฟส และสามารถปรับความเร็วของการหมุนตามค่าของ VR ที่ขาแอนะล็อก A0

ตัวอย่างที่ 8.8 โปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนครั้งละ 90 องศาตามการกดสวิตช์

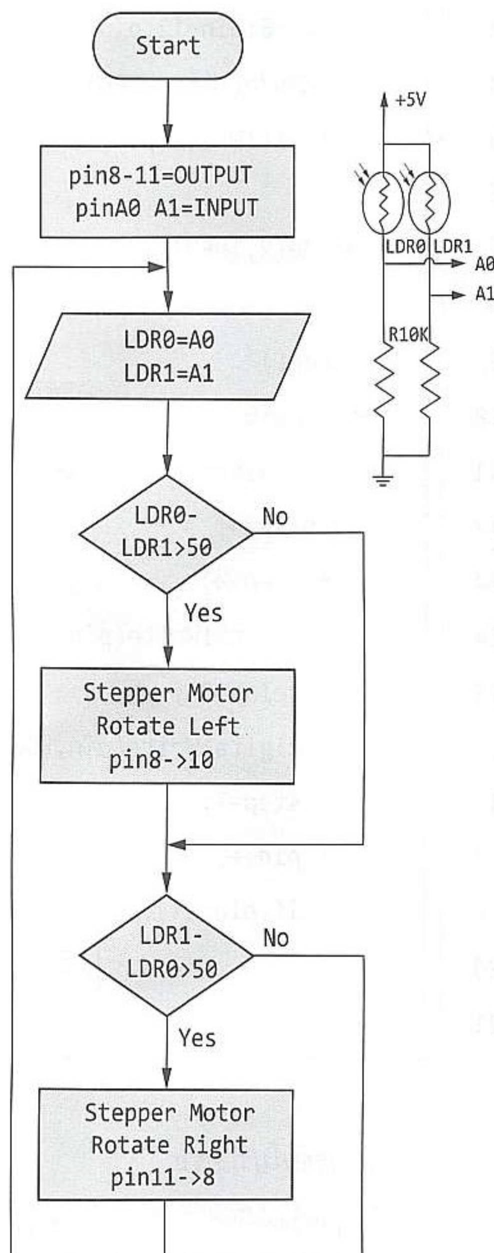
```
int pin;
void setup()
{ for(pin=8;pin<12;pin++)
   { pinMode(pin,OUTPUT);
     digitalWrite(pin,LOW);
   }
  pinMode(2,INPUT);
}
void loop()
{ int step=0;
  if(digitalRead(2)==HIGH)
    step=1024;
  while(step>0)
      { digitalWrite(pin,HIGH);
        delay(5);
        digitalWrite(pin,LOW);
        step--;
        pin++;
        if(pin>11) pin=8;
      }
}
```

ผลการรันโปรแกรม

เมื่อกดสวิตช์ที่ขา 2 ตัวแปร step = 1024 โปรแกรมจะวนรอบส่งข้อมูล HIGH หรือลอจิก 1 ออกทางขา 8, 9, 10 และ 11 เป็นจำนวน 1024 ครั้ง ทำให้มอเตอร์หมุน 90 องศา

ตัวอย่างที่ 8.9 โปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนตามแสง

```
int pin;
void setup()
{ for(pin=8;pin<12;pin++)
   { pinMode(pin,OUTPUT);
     digitalWrite(pin,LOW);
   }
  pinMode(A0,INPUT);
  pinMode(A1,INPUT);
}
void loop()
{  int rotate;
   int LDR0=analogRead(A0);
   int LDR1=analogRead(A1);
   if(LDR0-LDR1 > 50)
    {    pin++;
         if(pin>11) pin=8;
         digitalWrite(pin,HIGH);
         delay(20);
         digitalWrite(pin,LOW);
    }
  if(LDR1-LDR0 > 50)
    {    pin--;
         if(pin<8) pin=11;
         digitalWrite(pin,HIGH);
         delay(20);
         digitalWrite(pin,LOW);
    }
}
```



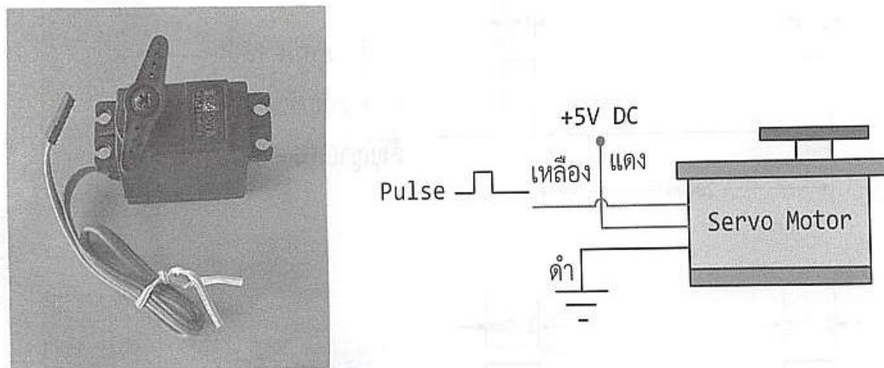
รูปที่ 8.15 โฟลว์ชาร์ตควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนตามแสง

ผลการรันโปรแกรม

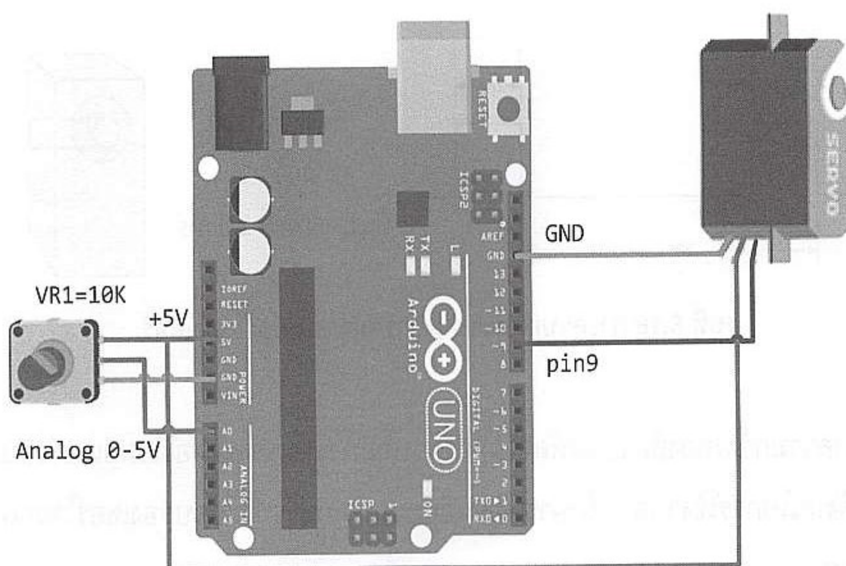
โปรแกรมวนรอบรอรับค่าแสงมาเก็บไว้ในตัวแปร LDR0 และ LDR1 ถ้าค่าของ LDR0 - LDR1 มากกว่า 50 หน่วย ให้สเต็ปเปอร์มอเตอร์หมุนไปทางซ้าย ถ้าค่าของ LDR1 - LDR0 มากกว่า 50 หน่วย ให้สเต็ปเปอร์มอเตอร์หมุนไปทางขวา

8.6 เซอร์โวมอเตอร์

เซอร์โวมอเตอร์ (Servo Motor) คือมอเตอร์ที่สามารถควบคุมความเร็ว ตำแหน่ง และอัตราเร่งได้ โดยเซอร์โวมอเตอร์จะมีส่วนของวงจรป้อนกลับเพื่อตรวจสอบการทำงาน มีลักษณะควบคุมแบบป้อนกลับ เซอร์โวมอเตอร์มีหลายชนิด ในหัวข้อนี้จะกล่าวถึง RC เซอร์โวมอเตอร์ ซึ่งเป็นเซอร์โวมอเตอร์ชนิดหนึ่ง มีขนาดเล็ก น้ำหนักเบา มีแรงบิดสูง ตัวถังเป็นสี่เหลี่ยมสามารถติดตั้งได้ง่าย RC เซอร์โวมอเตอร์จะมีสายสัญญาณ 3 เส้น ประกอบไปด้วยสายสัญญาณไฟ กราวด์ และสัญญาณพัลส์ควบคุมการทำงานของเซอร์โวมอเตอร์ แสดงดังรูปที่ 8.16



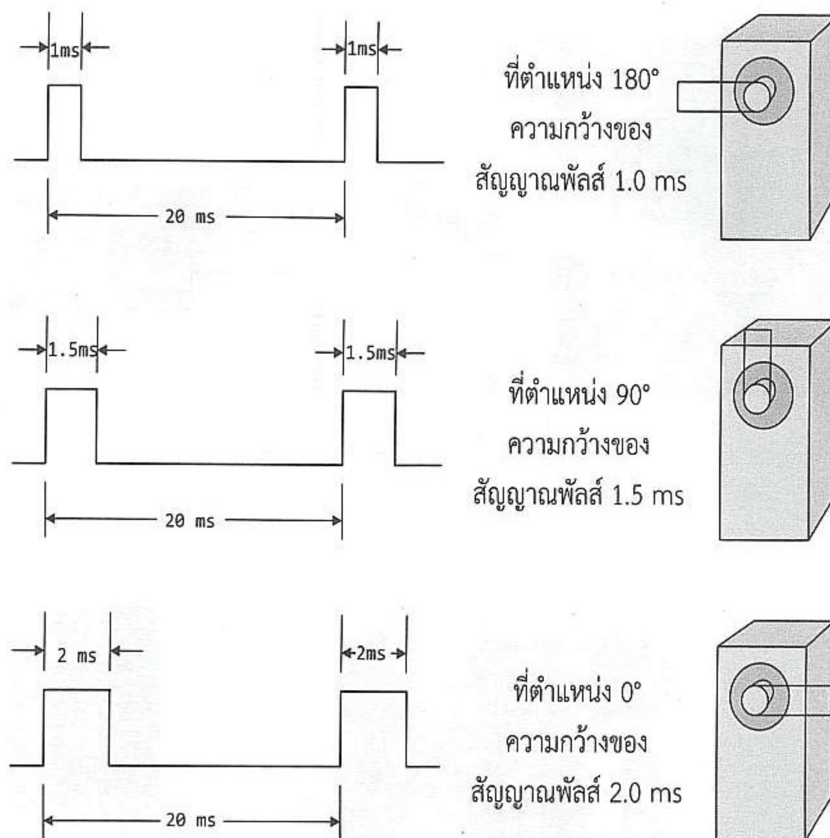
รูปที่ 8.16 เซอร์โวมอเตอร์และขาสัญญาณ



รูปที่ 8.17 การเชื่อมต่อ Arduino กับเซอร์โวมอเตอร์

8.7 การควบคุมเซอร์โวมอเตอร์

การควบคุมตำแหน่งการหมุนของเซอร์โวมอเตอร์สามารถควบคุมได้โดยปรับความกว้างของสัญญาณพัลส์ ถ้าส่งสัญญาณพัลส์ที่มีความกว้างขนาด 1 ms จะทำให้เซอร์โวมอเตอร์หมุนมาตำแหน่งซ้ายสุดหรือที่ตำแหน่ง 180 องศา ถ้าส่งสัญญาณพัลส์ 1.5 ms เซอร์โวมอเตอร์จะหมุนมาที่ตำแหน่งตรงกลางหรือที่ตำแหน่ง 90 องศา และถ้าส่งสัญญาณพัลส์ 2 ms เซอร์โวมอเตอร์จะหมุนมาทางขวาสุดหรือที่ตำแหน่ง 0 องศา และต้องส่งสัญญาณพัลส์ให้ทุก ๆ 20 ms เพื่อรักษาตำแหน่งการหมุนของเซอร์โวมอเตอร์ ดังแสดงในรูปที่ 8.18

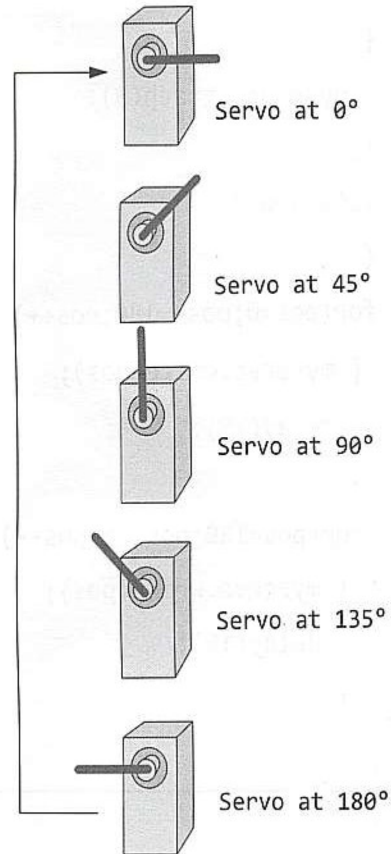


รูปที่ 8.18 การควบคุมตำแหน่งการหมุนของเซอร์โวมอเตอร์

ค่าความกว้างของสัญญาณพัลส์ในการควบคุมการหมุนของเซอร์โวมอเตอร์เป็นเพียงค่าประมาณ ดังนั้นในการใช้งานต้องศึกษารายละเอียดและเอกสารประกอบของเซอร์โวมอเตอร์ที่นำมาใช้งานด้วย

ตัวอย่างที่ 8.10 โปรแกรมควบคุมเซอร์โวมอเตอร์ให้หมุน 0, 45, 90, 135, 180 องศา

```
#include <Servo.h>
Servo myservo;
void setup()
{
  myservo.attach(9);
}
void loop()
{
  myservo.write(0);
  delay(1000);
  myservo.write(45);
  delay(1000);
  myservo.write(90);
  delay(1000);
  myservo.write(135);
  delay(1000);
  myservo.write(180);
  delay(1000);
}
```

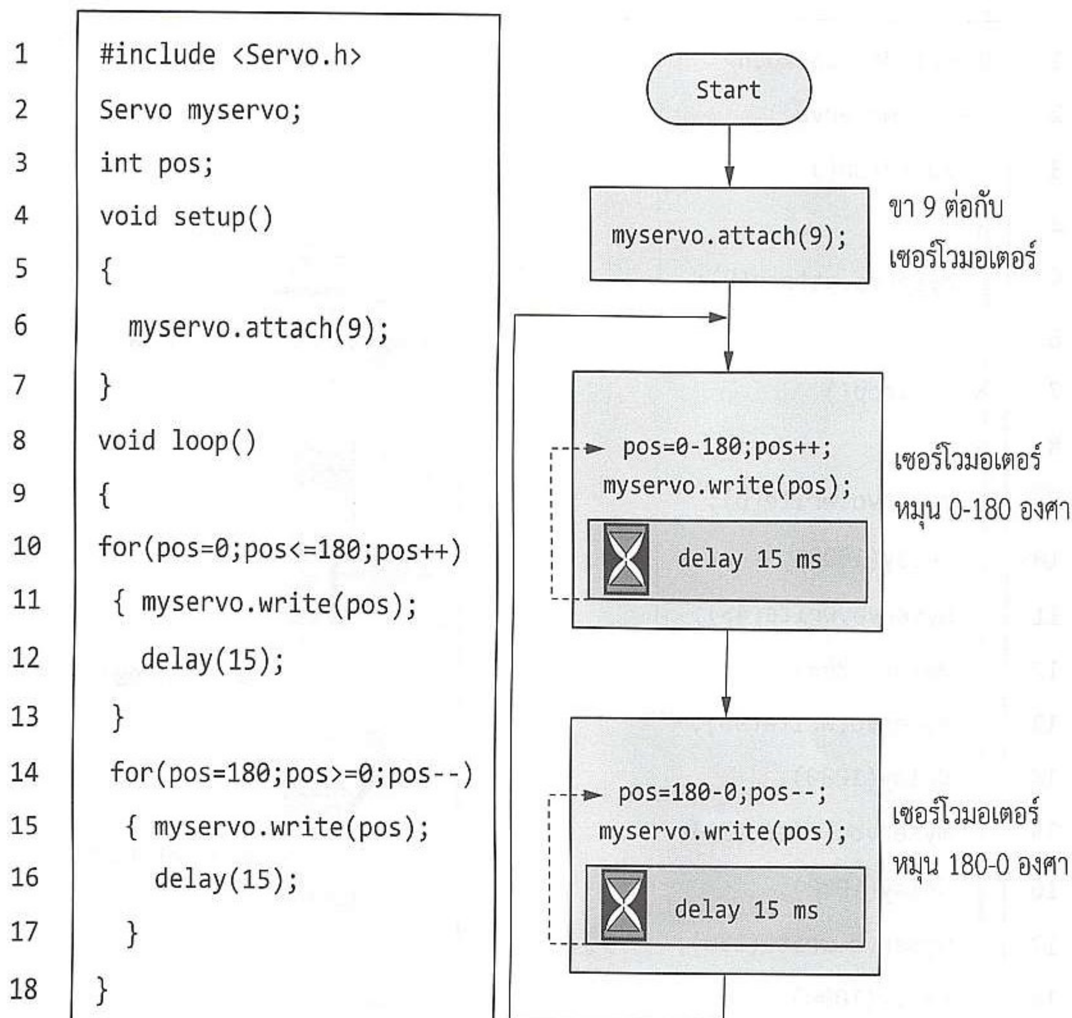


รูปที่ 8.19 การหมุนของเซอร์โวมอเตอร์ที่ตำแหน่ง 0, 45, 90, 135 และ 180 องศา

ผลการรันโปรแกรม

โปรแกรมจะวนรอบควบคุมให้เซอร์โวมอเตอร์หมุนไปเรื่อย ๆ ที่ตำแหน่ง 0, 45, 90, 135 และ 180 องศา โดยฟังก์ชัน myservo.write(); จะควบคุมให้สเต็ปเปอร์มอเตอร์หมุนไปตามมุมที่ป้อนได้ตั้งแต่ 0 ถึง 180 องศา

ตัวอย่างที่ 8.11 โปรแกรมควบคุมเซอร์โวมอเตอร์ให้หมุนซ้ายและขวา



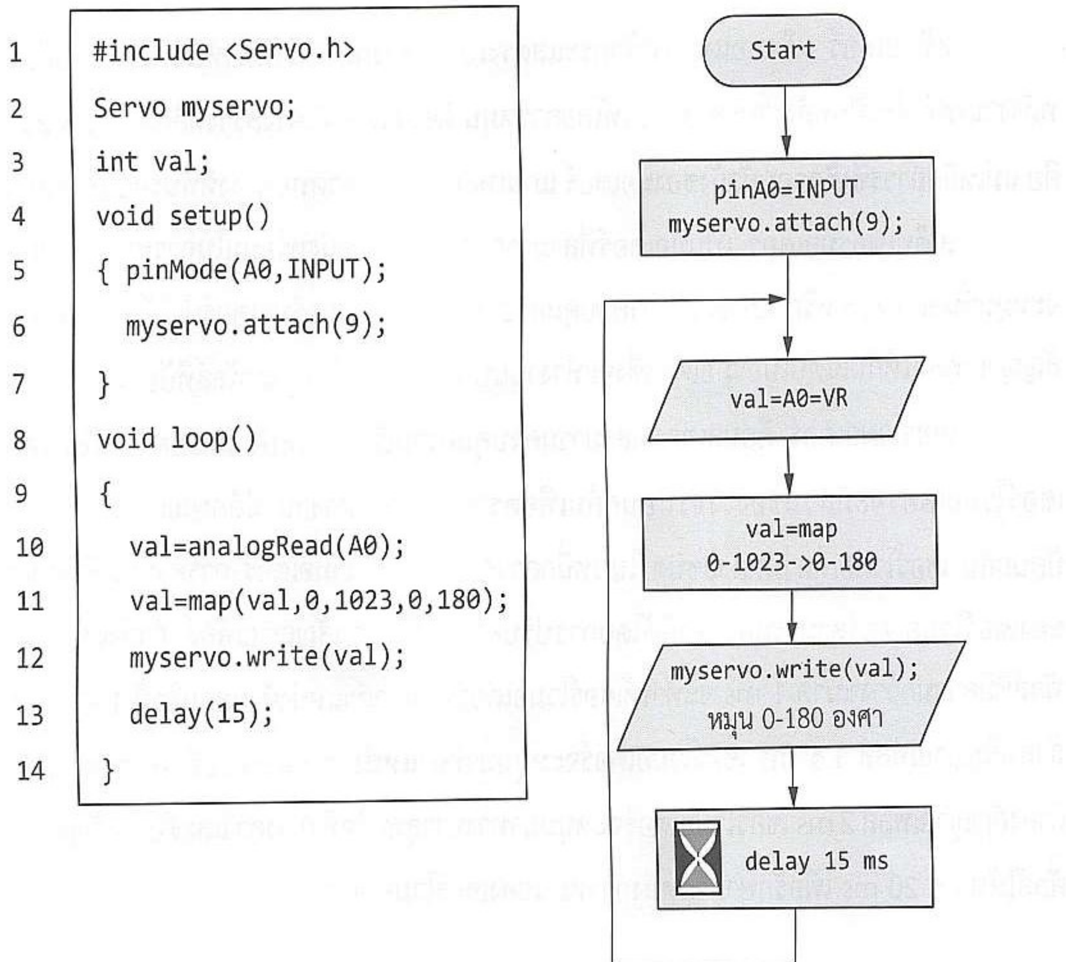
```
#include <Servo.h>
Servo myservo;
int pos;
void setup()
{
  myservo.attach(9);
}
void loop()
{
for(pos=0;pos<=180;pos++)
 { myservo.write(pos);
   delay(15);
 }
 for(pos=180;pos>=0;pos--)
  { myservo.write(pos);
    delay(15);
  }
}
```


รูปที่ 8.20 โฟลว์ชาร์ตควบคุมการหมุนซ้ายและขวาของเซอร์โวมอเตอร์

ผลการรันโปรแกรม

โปรแกรมจะควบคุมให้เซอร์โวมอเตอร์หมุนจาก 0-180 องศา แล้วหมุนกลับ 180-0 องศา วนรอบหมุนไปกลับ

ตัวอย่างที่ 8.12 โปรแกรมควบคุมเซอร์โวมอเตอร์ให้หมุนตามค่า VR



```
#include <Servo.h>
Servo myservo;
int val;
void setup()
{ pinMode(A0,INPUT);
  myservo.attach(9);
}
void loop()
{
  val=analogRead(A0);
  val=map(val,0,1023,0,180);
  myservo.write(val);
  delay(15);
}
```


รูปที่ 8.21 โฟลว์ชาร์ตควบคุมเซอร์โวมอเตอร์ตามค่า VR

ผลการรันโปรแกรม

โปรแกรมจะวนรอบรอรับค่าความต้านทานแบบปรับค่าได้ (VR) จากขา A0 มาเก็บไว้ที่ตัวแปร val จากนั้นทำการแปลงค่าจากช่วง 0-1023 ให้อยู่ในช่วง 0-180 แล้วส่งค่าให้ฟังก์ชัน myservo.write(); เพื่อควบคุมเซอร์โวมอเตอร์ให้หมุน 0-180 องศาตามค่าตัวแปร val

8.8 สรุป

ดีซีมอเตอร์ หรือมอเตอร์ไฟฟ้ากระแสตรงเป็นเครื่องกลไฟฟ้าที่ทำหน้าที่ในการเปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานกล เคลื่อนที่โดยการหมุน ดีซีมอเตอร์มีโครงสร้างหลักสำคัญ 3 ส่วน คือ แม่เหล็กถาวรซึ่งติดรอบตัวถังของมอเตอร์ แกนเหล็ก และขดลวดทองแดงที่พันรอบแกนเหล็ก

สเต็ปเปอร์มอเตอร์ เป็นมอเตอร์ที่สามารถควบคุมตำแหน่งหรือมุมในการหมุนได้ ปกติจะหมุนขั้นละ 1.8, 5 หรือ 7.5 องศา การควบคุมการหมุนของสเต็ปเปอร์มอเตอร์ทำได้โดยการป้อนสัญญาณพัลส์ให้กับสเต็ปเปอร์มอเตอร์ซึ่งจะทำงานแบบลำดับตามสัญญาณพัลส์ที่ป้อนเข้า

เซอร์โวมอเตอร์ คือมอเตอร์ที่สามารถควบคุมความเร็ว ตำแหน่ง และอัตราเร่งได้ โดยเซอร์โวมอเตอร์จะมีส่วนของวงจรป้อนกลับเพื่อตรวจสอบการทำงาน มีลักษณะควบคุมแบบป้อนกลับ เซอร์โวมอเตอร์มีหลายชนิด ในบทนี้กล่าวถึง RC เซอร์โวมอเตอร์ การควบคุมตำแหน่งของเซอร์โวมอเตอร์สามารถควบคุมได้โดยการปรับความกว้างของสัญญาณพัลส์ ถ้าส่งสัญญาณพัลส์ที่มีความกว้างขนาด 1 ms จะทำให้เซอร์โวมอเตอร์หมุนมาตำแหน่งซ้ายสุดหรือที่ 180 องศา ถ้าส่งสัญญาณพัลส์ 1.5 ms เซอร์โวมอเตอร์จะหมุนมาที่ตำแหน่งตรงกลางหรือที่ 90 องศา และถ้าส่งสัญญาณพัลส์ 2 ms เซอร์โวมอเตอร์จะหมุนมาทางขวาสุดหรือที่ 0 องศา และต้องส่งสัญญาณพัลส์ให้ทุก ๆ 20 ms เพื่อรักษาตำแหน่งการหมุนของเซอร์โวมอเตอร์

คำถามท้ายบทที่ 8

1. จงเขียนโปรแกรมควบคุมดีซีมอเตอร์ให้หมุน 2 วินาทีและหยุด 3 วินาที สลับกันไปมา
2. จงเขียนโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์แบบครึ่งขั้น
3. จงเขียนโปรแกรมควบคุมการหมุนของสเต็ปเปอร์มอเตอร์ให้หมุนครั้งละ 90 องศาตามการกดสวิตช์
4. จงเขียนโปรแกรมควบคุมสเต็ปเปอร์มอเตอร์ให้หมุนซ้ายขวาตามการกดสวิตช์
5. จงอธิบายหลักการควบคุมการหมุนของเซอร์โวมอเตอร์
6. จงเขียนโปรแกรมควบคุมเซอร์โวมอเตอร์ให้หมุนครั้งละ 10 องศา
7. จงเขียนโปรแกรมควบคุมเซอร์โวมอเตอร์ให้หมุนตามความเข้มของแสง โดยใช้ LDR1 และ LDR2 เป็นตัวตรวจวัดความเข้มของแสง แล้วให้เซอร์โวมอเตอร์หมุนไปตามทิศทางของแสง

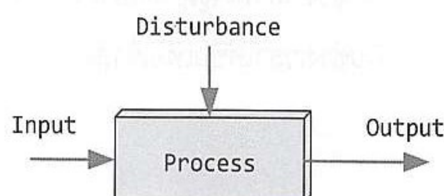
9

ระบบควบคุมแบบ PID

ระบบควบคุมแบบดั้งเดิมเป็นระบบที่ควบคุมด้วยมือ เช่นใช้มือในการปิดเปิดน้ำ ซึ่งใช้มาตั้งแต่ในอดีต ต่อมามีการปรับปรุงและพัฒนาระบบควบคุมเพื่อบังคับการทำงานของกระบวนการบางอย่างให้มีประสิทธิภาพ ตัวอย่างเช่น การใช้ลูกลอยควบคุมระดับน้ำในถังเพื่อปิดน้ำเมื่อน้ำเต็มถังทำให้ประหยัดน้ำ อำนวยความสะดวก และประหยัดเวลาในการตรวจสอบระดับน้ำ จุดเริ่มต้นของการประยุกต์ใช้ระบบควบคุมแบบป้อนกลับในวงการอุตสาหกรรมเกิดขึ้นในปี ค.ศ. 1788 เมื่อเจมส์ วัตต์ (James Watt) ได้สร้างเครื่องจักรไอน้ำ ซึ่งใช้แบบจำลองทางคณิตศาสตร์มาวิเคราะห์และออกแบบระบบควบคุมความเร็วของเครื่องจักรไอน้ำให้ได้ความเร็วตามที่ต้องการ

9.1 พื้นฐานของระบบควบคุม

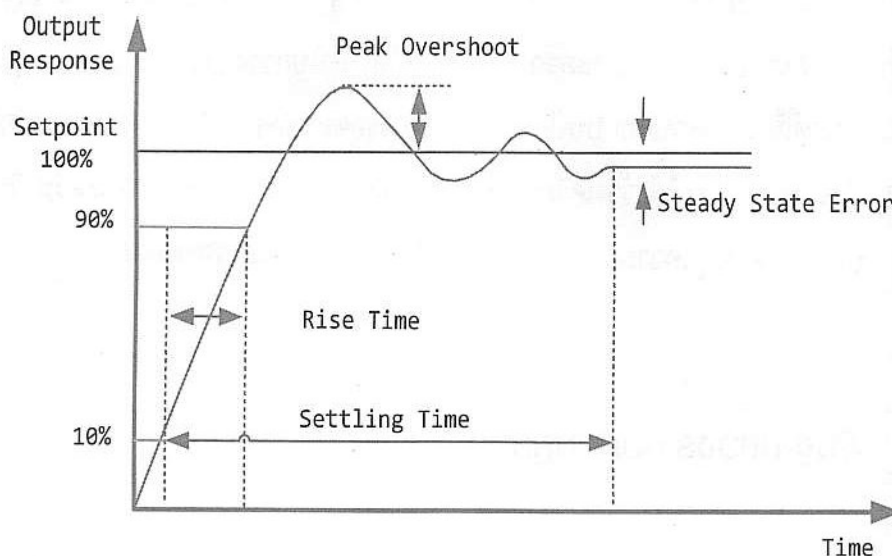
ระบบควบคุม (Control System) หมายถึง ระบบการบังคับ ควบคุม สั่งการเพื่อควบคุมกระบวนการให้ทำงานตามวัตถุประสงค์ที่ต้องการ ซึ่งระบบควบคุมที่ดีควรมีเสถียรภาพ สามารถควบคุมสัญญาณเอาต์พุตให้เป็นไปตามค่าที่ตั้งไว้ มีการตอบสนองรวดเร็วทันเวลา สัญญาณเอาต์พุตไม่แกว่งและมีค่าผิดพลาดน้อย



รูปที่ 9.1 องค์ประกอบของระบบควบคุม

Input	คือ สัญญาณหรืออุปกรณ์อินพุต
Process	คือ กระบวนการทำงานของระบบ
Output	คือ สัญญาณเอาต์พุต
Disturbance	คือ สัญญาณรบกวน

จุดมุ่งหมายของการควบคุม คือ การพยายามควบคุมค่าสัญญาณเอาต์พุตให้มีค่าเป็นไปตามค่าที่ตั้งไว้อยู่ตลอดเวลา แม้ว่าจะเกิดการเปลี่ยนแปลงขึ้นในระบบ ตัวอย่างเช่น ต้องการควบคุมให้เตาเผาเซรามิกทำงานที่อุณหภูมิ 800 องศา แม้ว่าโหลดจะมีการเปลี่ยนแปลง เช่น ใส่เซรามิกจำนวนมากหรือน้อย หรือเปลี่ยนค่าอุณหภูมิในการเผา ระบบควบคุมที่ดีต้องพยายามควบคุมอุณหภูมิให้เป็นไปตามค่าที่กำหนดไว้ สิ่งที่ต้องพิจารณาคือกราฟผลตอบสนองของระบบควบคุม ซึ่งจำเป็นต้องเข้าใจตัวแปรหรือพารามิเตอร์ต่าง ๆ ดังนี้



รูปที่ 9.2 กราฟผลตอบสนองเอาต์พุตของระบบควบคุม

Setpoint	คือ ค่าเป้าหมายที่ตั้งไว้หรือกำหนดไว้ เช่น ตั้งอุณหภูมิเตาเผาเซรามิกที่ 800 องศา
Rise Time	คือ ช่วงเวลาที่สัญญาณเอาต์พุตเปลี่ยนจาก 10% เป็น 90% ซึ่งเป็นช่วงเวลาไต่ขึ้นเพื่อเข้าสู่ค่าเป้าหมายที่ตั้งไว้

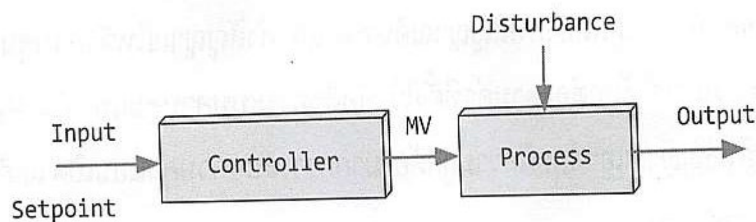
Peak Overshoot คือ สัญญาณเอาต์พุตสูงสุดที่เกินค่าเป้าหมายที่ตั้งไว้ เช่น ตั้งไว้ที่ 800 องศา แต่ได้อุณหภูมิ 900 องศา เกิด Peak Overshoot จะทำให้เซรามิกเสียหายหรือไม่ ถ้าเสียหายก็ต้องปรับค่าลง

Settling Time คือ ช่วงเวลาที่สัญญาณเอาต์พุตเป็น 10% จนถึงสัญญาณเอาต์พุตคงตัวไม่มีการเปลี่ยนแปลง

Steady State Error คือ ค่าผิดพลาดเมื่อเข้าสู่สภาวะคงตัว

Disturbance คือ สัญญาณรบกวน เช่น ในฤดูร้อนหรือฤดูหนาวระบบอาจทำงานมากขึ้น

9.2 ระบบควบคุมแบบเปิด



รูปที่ 9.3 ระบบควบคุมแบบเปิด

ระบบควบคุมแบบเปิด (Open Loop Control) เป็นระบบควบคุมแบบทิศทางเดียว คือส่งสัญญาณอินพุตไปยังตัวควบคุมเพื่อควบคุมกระบวนการ โดยไม่มีการป้อนกลับของสัญญาณเอาต์พุตเพื่อมาเปรียบเทียบหรือตรวจสอบความถูกต้องกับสัญญาณอินพุต ข้อดีของระบบควบคุมแบบเปิดคือ เป็นระบบควบคุมที่ง่าย ไม่ซับซ้อน ทนทาน และมีราคาถูก

ตัวอย่างเช่น การเปิดพัดลมเป็นระบบควบคุมแบบเปิด ไม่มีการป้อนกลับของสัญญาณเอาต์พุตคือความเร็วของลมหรือความเย็นของอากาศ เพื่อมาเปรียบเทียบหรือตรวจสอบความถูกต้อง ดังนั้นผู้ใช้งานต้องปรับอินพุตหรือกดสวิตช์ 1, 2 และ 3 ตามความต้องการของผู้ใช้งาน

Setpoint คือ ต้องการให้อากาศเย็นหรือให้อากาศเคลื่อนที่ สั่งงานผ่านอุปกรณ์อินพุตโดยการกดสวิตช์ 1, 2 และ 3

Controller คือ วงจรควบคุม ส่วนใหญ่จะเป็นวงจรไฟฟ้า

Manipulate Variable หรือ MV คือสัญญาณควบคุมหรือสวิตช์ควบคุมให้มอเตอร์ทำงาน

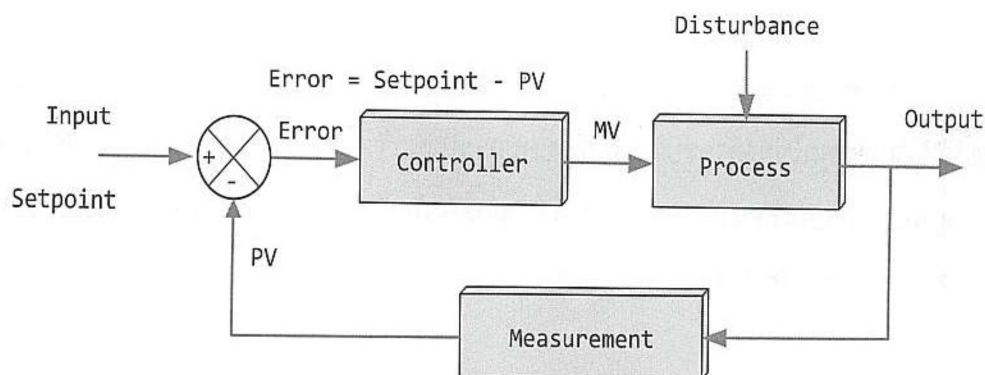
Process คือ กระบวนการที่ทำให้อากาศเย็นหรือมอเตอร์หมุน

Output คือ ความเย็นของอากาศหรือความเร็วในการเคลื่อนที่ของอากาศเพื่อระบายความร้อน

Disturbance คือ สัญญาณรบกวน เช่น ในฤดูร้อนอุณหภูมิจะสูงทำให้เปิดพัดลมแล้วอาจไม่เย็นอย่างที่ต้องการ

9.3 ระบบควบคุมแบบปิด

ระบบควบคุมแบบปิด (Closed Loop Control) หรือระบบควบคุมแบบป้อนกลับ (Feedback Control) เป็นระบบควบคุมที่มีการป้อนกลับของสัญญาณเอาต์พุตเพื่อนำมาเปรียบเทียบหรือตรวจสอบความถูกต้องกับสัญญาณอินพุต แล้วส่งสัญญาณให้ตัวควบคุมเพื่อปรับการทำงานของกระบวนการให้ถูกต้องตามค่าที่ตั้งไว้ ข้อดีของระบบควบคุมแบบปิด คือเป็นระบบที่สามารถควบคุมให้สัญญาณเอาต์พุตมีความถูกต้องมากกว่าระบบควบคุมแบบเปิด แต่ก็มีการทำงานที่ซับซ้อนและมีราคาแพง



รูปที่ 9.4 ระบบควบคุมแบบปิด

ตัวอย่างเช่น การเปิดเครื่องปรับอากาศเป็นระบบการควบคุมแบบปิด มีการป้อนกลับของสัญญาณเอาต์พุตคือความเย็นหรืออุณหภูมิของอากาศเพื่อมาเปรียบเทียบหรือตรวจสอบความถูกต้องกับค่าที่ตั้งไว้ แล้วให้ตัวควบคุมสั่งงานกระบวนการปรับอุณหภูมิให้ได้ตามค่าที่ตั้งไว้

Setpoint คือ อุณหภูมิที่ต้องการ เช่น ต้องการให้อุณหภูมิเป็น 25 องศาก็สั่งงานผ่านอุปกรณ์อินพุตหรือรีโมทคอนโทรล

Controller คือ วงจรควบคุม ส่วนใหญ่จะเป็นไมโครคอนโทรลเลอร์และวงจรไฟฟ้า ทำหน้าที่ควบคุมกระบวนการที่ทำให้อุณหภูมิลดลงหรือทำให้อากาศในห้องเย็นลง

Manipulate Variable หรือ MV คือ สัญญาณควบคุมให้คอมเพรสเซอร์ทำงาน

Process คือ กระบวนการที่ทำให้อากาศเย็นหรือคอมเพรสเซอร์ทำงาน

Output คือ อุณหภูมิหรือความเย็นของอากาศ

Measurement คือ ตัววัดอุณหภูมิซึ่งติดอยู่ในเครื่องปรับอากาศ

Process Variable คือ ค่าของอุณหภูมิที่วัดได้ เช่น 28 องศา

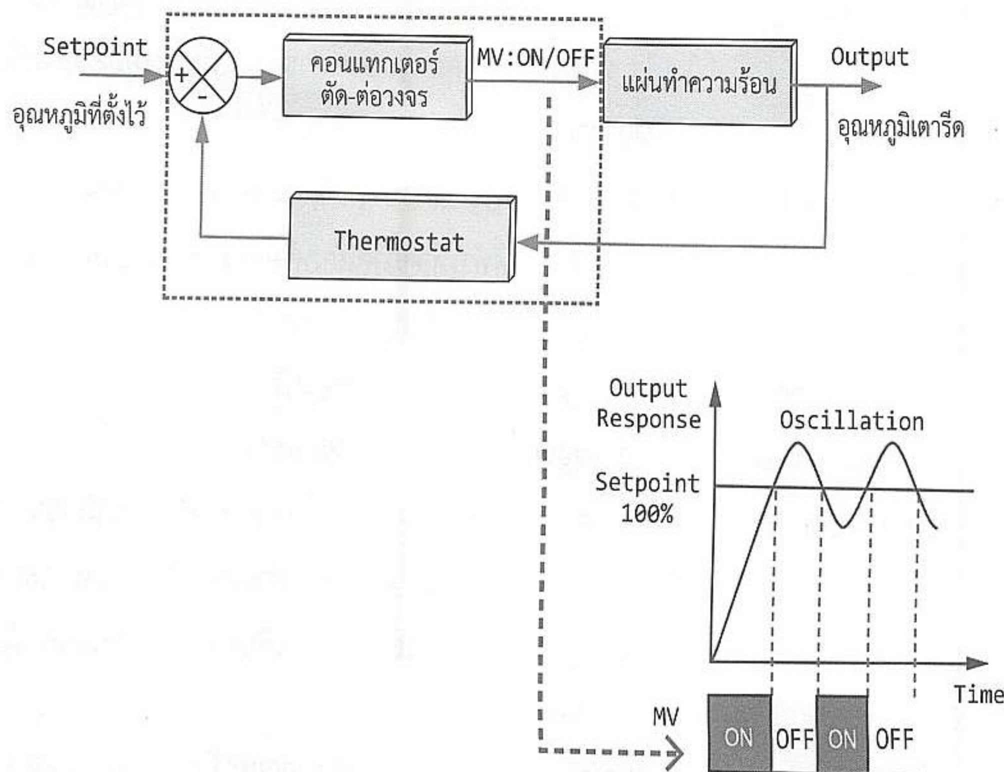
Error = Setpoint - Process Variable หรือ PV คือ ค่าผิดพลาดของอุณหภูมิ เช่น ตั้งไว้ที่ 25 องศา แต่วัดได้ 28 องศา เกิดค่าผิดพลาด -3 องศา ระบบจะส่งสัญญาณให้คอนโทรลเลอร์รับรู้เพื่อควบคุมกระบวนการให้เป็นไปตามค่าที่กำหนดไว้

Disturbance คือสัญญาณรบกวน เช่น ในฤดูร้อนอุณหภูมิภายนอกจะสูงทำให้เย็นช้า การเข้าถึงค่า Setpoint จึงใช้เวลานาน

9.4 ระบบควบคุมแบบเปิด-ปิด

ระบบควบคุมแบบเปิด-ปิด (ON-OFF Control) เป็นระบบควบคุมให้เอาต์พุตทำงานเพียง 2 สภาวะเท่านั้น คือ เปิดและปิด หรือ ON และ OFF เป็นการควบคุมแบบ 2 ตำแหน่งอย่างง่าย มีราคาถูก สัญญาณเอาต์พุตมีการแกว่ง เกิดการเปลี่ยนแปลงไม่คงที่

ตัวอย่างเช่น การควบคุมการทำงานของเตารีด เมื่อเสียบปลั๊ก เตารีดจะเริ่มทำงาน คอนแทกเตอร์ต่อวงจร ขดลวดความร้อนเริ่มทำงานทำให้อุณหภูมิค่อย ๆ เพิ่มขึ้น จนเมื่อถึงค่าที่ตั้งไว้ เทอร์โมสตัต (Thermostat) จะตัดวงจรทำให้ขดลวดหยุดทำงาน จากนั้นสัญญาณเอาต์พุตหรือความร้อนจากเตารีดจะลดลง เอาต์พุตของสัญญาณเกิดการแกว่งไม่คงที่ตามการเปิด-ปิดของคอนแทกเตอร์ ระบบควบคุมแบบเปิด-ปิดส่วนใหญ่จะใช้กับเครื่องใช้ไฟฟ้าภายในบ้าน เนื่องจากเอาต์พุตที่แกว่งไม่มีผลต่อระบบหรือผู้ใช้งาน เช่น เตารีดจะร้อนมากหรือน้อยก็สามารถรีดผ้าได้ น้ำจากกาต้มน้ำจะร้อนมากหรือน้อยก็สามารถชงกาแฟได้

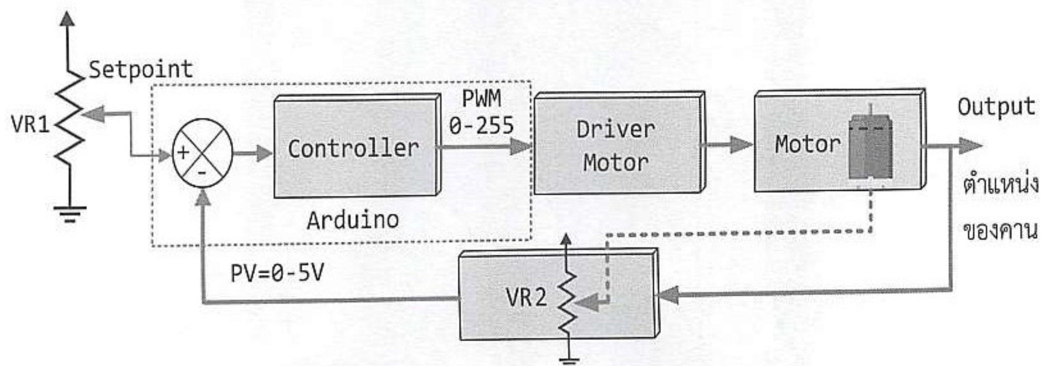


รูปที่ 9.5 ระบบควบคุมแบบเปิด-ปิดและผลตอบสนองของเอาต์พุต

Setpoint	คือ อุณหภูมิที่ต้องการ เช่น ร้อนน้อย ร้อนปานกลาง หรือร้อนมาก
Controller	คือ ตัวคอนแทกเตอร์ที่ทำการตัด-ต่อวงจร
Manipulated Variable หรือ MV	คือ สัญญาณควบคุม ON หรือ OFF ที่ได้จากคอนแทกเตอร์
Process	คือ กระบวนการของขดลวดทำความร้อนเพื่อทำให้เกิดความร้อน
Measurement	คือ เทอร์โมสตัต หรือตัววัดอุณหภูมิซึ่งอยู่ในเตารีด
Output	คือ อุณหภูมิหรือความร้อนที่ได้
Oscillation	คือ การแกว่งของสัญญาณเอาต์พุตหรืออุณหภูมิของเตารีด

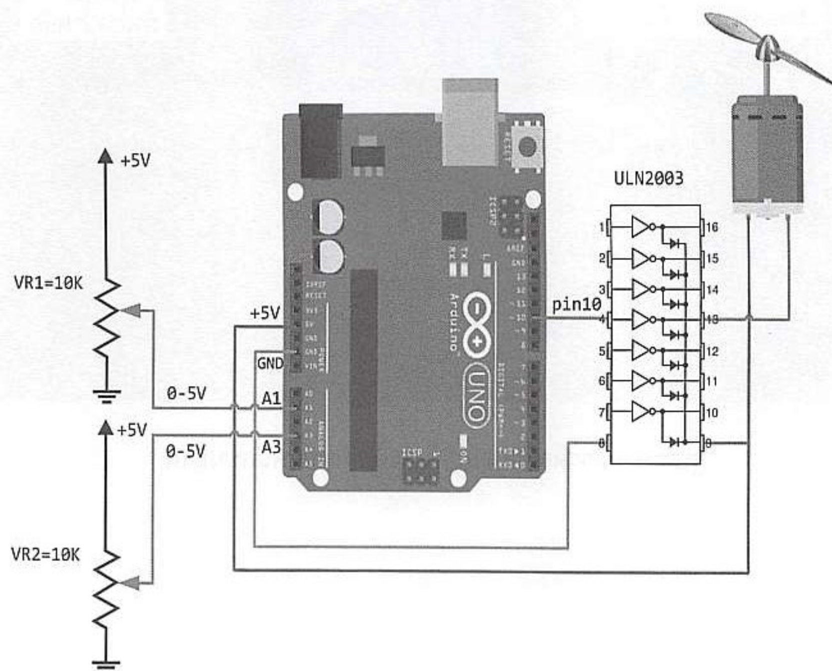
9.5 การควบคุมตำแหน่งของคาน

การสร้างชุดควบคุมตำแหน่งของคาน จะทำให้เห็นผลตอบสนองการทำงานของระบบควบคุมได้อย่างเป็นรูปธรรม เช่น ถ้าระบบมีการควบคุมตำแหน่งแบบเปิด-ปิด คานก็จะแกว่งขึ้นลงไปมาหรือไม่เสถียร หากเป็นการควบคุมแบบ PID (จะกล่าวถึงรายละเอียดในหัวข้อที่ 9.8) คานก็จะรักษาตำแหน่งที่ตั้งไว้ ไม่แกว่ง หรือมีความเสถียรมากกว่าการควบคุมแบบเปิด-ปิด



รูปที่ 9.6 บล็อกไดอะแกรมการควบคุมตำแหน่งของคาน

ชุดทดลองควบคุมตำแหน่งจะมี VR1 เป็นตัวปรับค่าตำแหน่งที่ต้องการ (Setpoint) มี VR2 เป็นอุปกรณ์วัดตำแหน่งของคาน แล้วส่งสัญญาณให้ไมโครคอนโทรลเลอร์ Arduino เข้าที่ขา A1 และ A3 เพื่อทำการประมวลผล จากนั้นส่งสัญญาณ PWM ออกทางขา 10 เพื่อขยายสัญญาณแล้วส่งไปควบคุมมอเตอร์ให้หมุนไปตามตำแหน่งที่กำหนด

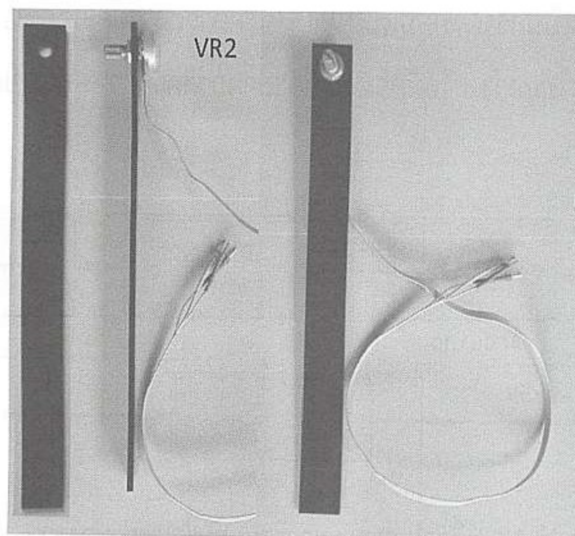


รูปที่ 9.7 วงจรการควบคุมตำแหน่งของคาน

9.6 ขั้นตอนการสร้างชุดควบคุมตำแหน่งของคาน

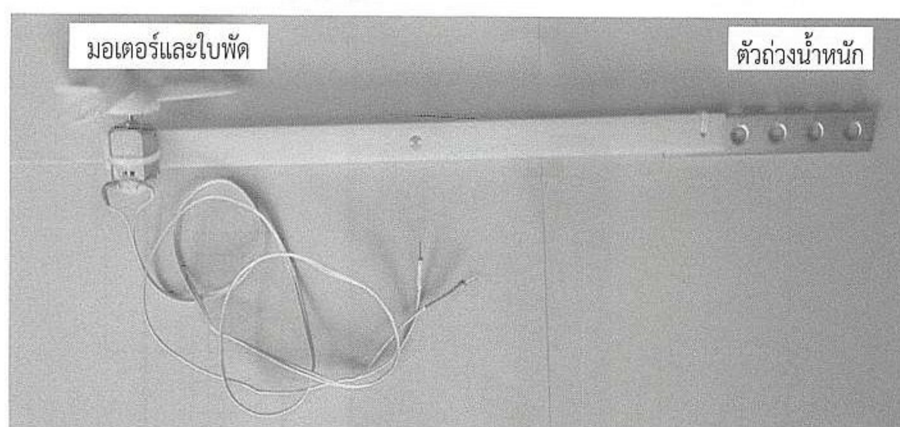
การสร้างชุดควบคุมตำแหน่งของคาน มีขั้นตอนดังนี้

1. ติดตั้งความต้านทานแบบปรับค่าได้หรือ VR2 กับฐานพลาสติก



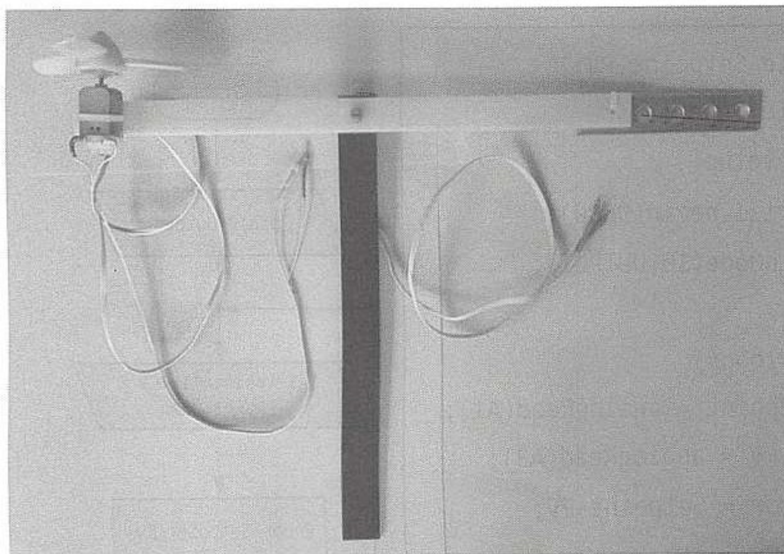
รูปที่ 9.8 ติดตั้ง VR2 เข้ากับฐานพลาสติก

2. ติดตั้งมอเตอร์และใบพัดเข้ากับคานที่ด้านซ้าย ส่วนด้านขวายึดด้วยตัวถ่วงน้ำหนักหรือดินน้ำมัน



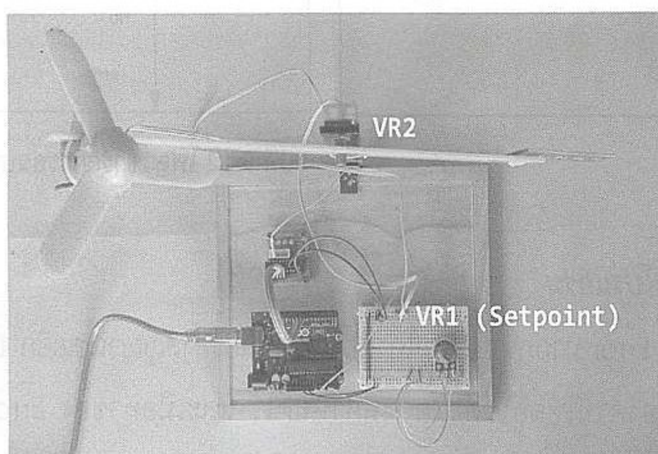
รูปที่ 9.9 การติดตั้งมอเตอร์และตุ้มถ่วงน้ำหนักบนคาน

3. ติดตั้งคานเข้ากับฐานพลาสติก



รูปที่ 9.10 ติดตั้งคานเข้ากับฐานพลาสติก

4. ติดตั้งชุดคานเข้ากับฐานไม้หรือโต๊ะ ต่อสายสัญญาณ VR1, VR2 วงจรขยายสัญญาณและมอเตอร์ ตามวงจรรูปที่ 9.7

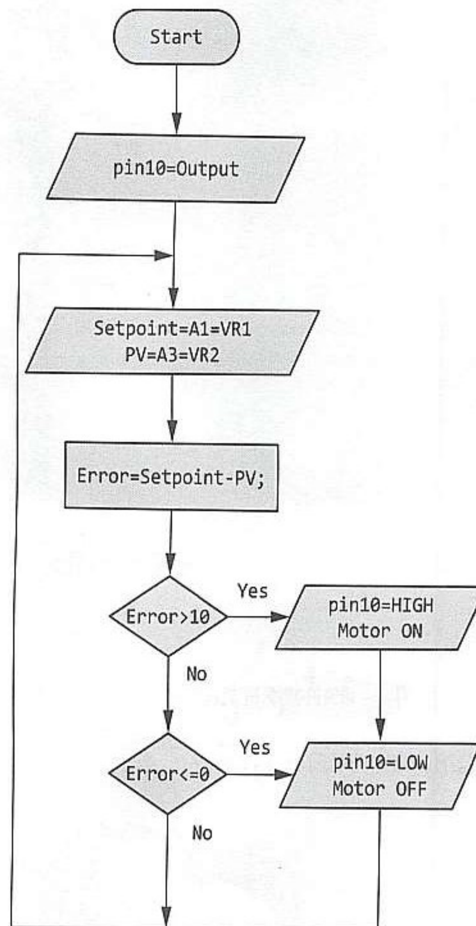


รูปที่ 9.11 ชุดควบคุมตำแหน่งของคาน

ในการทดลองให้ลองปรับค่าตัวถ่วงน้ำหนักหรือดินน้ำมันให้เหมาะสม คือทำให้คานเอียงในสภาวะที่ระบบไม่ทำงาน ถ้าระบบไม่สามารถควบคุมตำแหน่งได้ให้สลับขาสัญญาณไฟบวกและลบของ VR1, VR2 และมอเตอร์

ตัวอย่างที่ 9.1 โปรแกรมควบคุมตำแหน่งแบบเปิด-ปิด

```
double Setpoint,PV;
double Error;
void setup()
{ Serial.begin(9600);
   pinMode(10,OUTPUT);
}
void loop()
{ Setpoint = analogRead(A1);
      PV = analogRead(A3);
  Error = Setpoint-PV;
  if(Error>10)
    { digitalWrite(10,HIGH);
      Serial.print("ON");
    }
  if(Error<=0)
    { digitalWrite(10,LOW);
      Serial.print("OFF");
    }
}
```

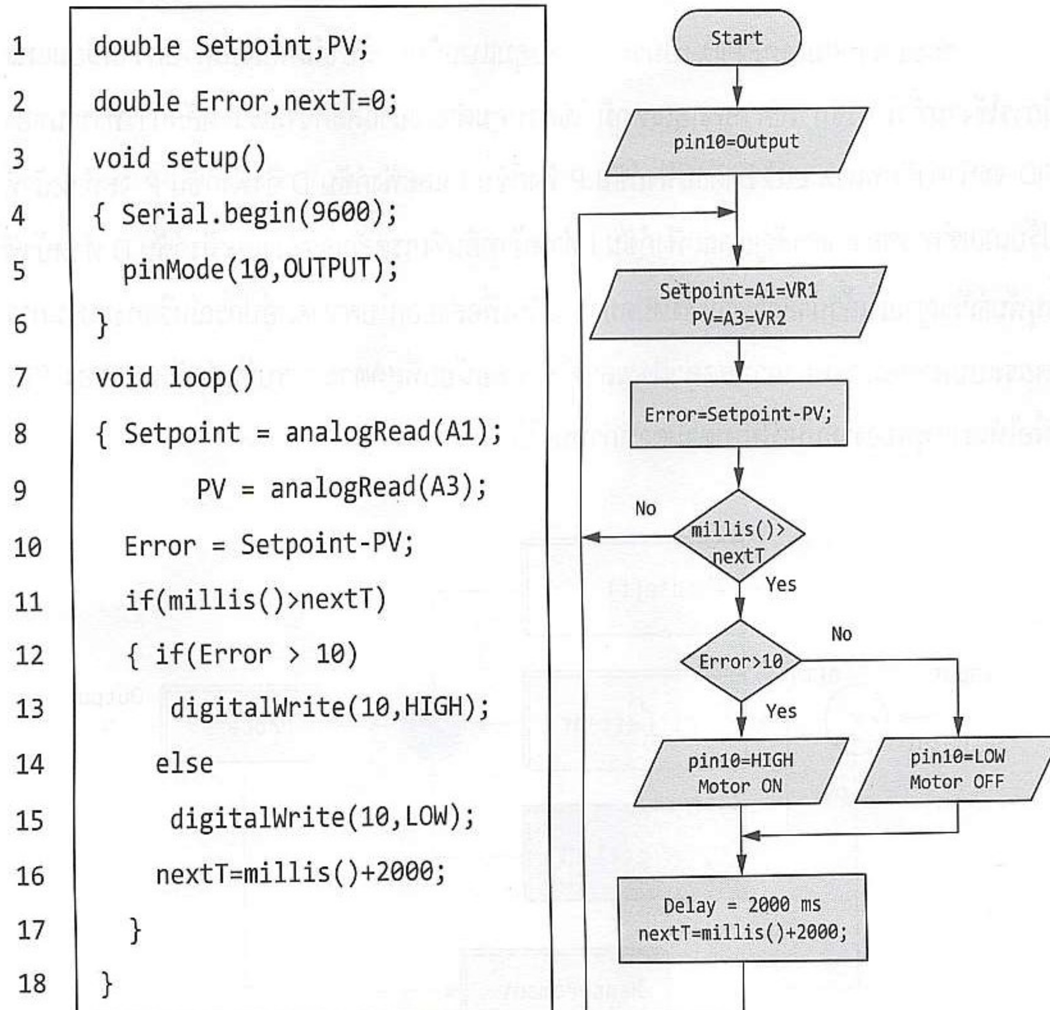


รูปที่ 9.12 โฟลว์ชาร์ตการควบคุมแบบเปิด-ปิด

ผลการรันโปรแกรม

ถ้าตัวแปร Error มากกว่า 10 ไมโครคอนโทรลเลอร์ส่งสัญญาณลอจิก 1 ให้มอเตอร์หมุนเพื่อปรับตำแหน่งของคานให้อยู่ในตำแหน่งที่กำหนดไว้ และเมื่อค่าของ Error น้อยกว่าหรือเท่ากับ 0 ไมโครคอนโทรลเลอร์ส่งสัญญาณลอจิก 0 ให้มอเตอร์หยุดหมุน ก็จะสามารถควบคุมตำแหน่งของคานได้ แต่จะทำให้เกิดการแกว่งของคาน ซึ่งเป็นค่าปกติของการควบคุมแบบเปิด-ปิด

ตัวอย่างที่ 9.2 การควบคุมตำแหน่งแบบเปิด-ปิดโดยการหน่วงเวลา



```
double Setpoint,PV;
double Error,nextT=0;
void setup()
{ Serial.begin(9600);
  pinMode(10,OUTPUT);
}
void loop()
{ Setpoint = analogRead(A1);
        PV = analogRead(A3);
  Error = Setpoint-PV;
  if(millis()>nextT)
  { if(Error > 10)
      digitalWrite(10,HIGH);
    else
      digitalWrite(10,LOW);
    nextT=millis()+2000;
  }
}
```

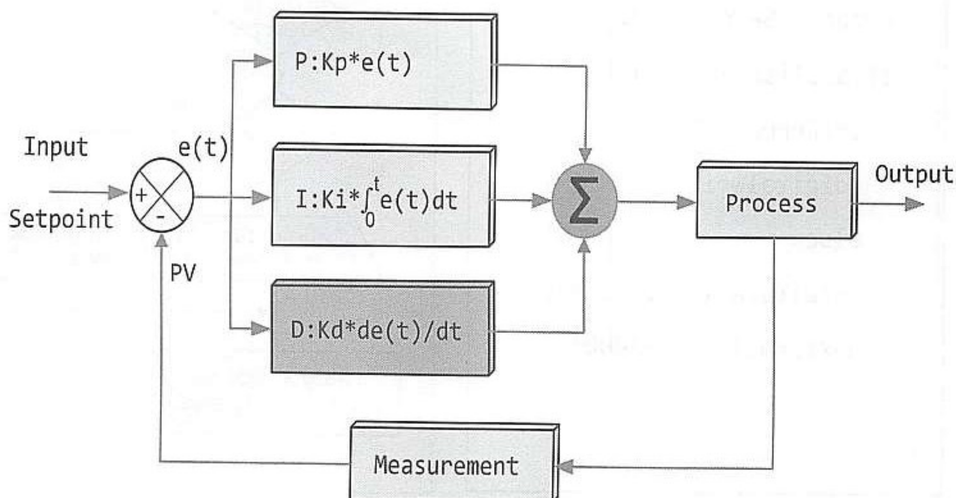

รูปที่ 9.13 โฟลว์ชาร์ตการควบคุมแบบเปิด-ปิด โดยการหน่วงเวลา

ผลการรันโปรแกรม

ถ้าตัวแปร Error มากกว่า 10 ไมโครคอนโทรลเลอร์ส่งสัญญาณลอจิก 1 เป็นเวลา 2000 ms หรือ 2 วินาทีให้มอเตอร์หมุนเพื่อปรับตำแหน่งของคานให้อยู่ในตำแหน่งที่ต้องการ ทำให้เกิดการแกว่งของคาน ซึ่งเป็นค่าปกติของการควบคุมแบบเปิด-ปิด

9.7 ระบบควบคุมแบบ PID

ระบบควบคุมแบบ PID เป็นระบบควบคุมแบบป้อนกลับชนิดหนึ่งที่ได้รับความนิยมและมีการใช้งานทั่วไป เช่น ใช้ควบคุมอุณหภูมิ ใช้ควบคุมตำแหน่งและความเร็ว หลักการทำงานของ PID จะนำค่าผิดพลาด e(t) มาผ่านฟังก์ชัน P ฟังก์ชัน I และฟังก์ชัน D ซึ่งฟังก์ชัน P จะทำหน้าที่ปรับเกณฑ์การขยายของสัญญาณ ฟังก์ชัน I ทำหน้าที่อินทิเกรตสัญญาณ และฟังก์ชัน D ทำหน้าที่อนุพันธ์สัญญาณ แล้วนำสัญญาณทั้งหมดมารวมกันเพื่อส่งออกไปควบคุมอุปกรณ์หรือกระบวนการ โดยระบบควบคุมจะพยายามลดค่าผิดพลาดให้เหลือน้อยที่สุดตามการปรับค่าตัวแปรของ PID เพื่อให้เอาต์พุตของระบบเป็นไปตามค่าที่กำหนดไว้



รูปที่ 9.14 ระบบควบคุมแบบ PID

1) **การควบคุมแบบ P หรือฟังก์ชัน P (Proportional)** เป็นการขยายแบบสัดส่วน ทำหน้าที่ปรับอัตราการขยายของสัญญาณความผิดพลาด ซึ่งมีตัวแปร Kp เป็นตัวคูณในการปรับค่าเกณฑ์การขยายสัดส่วน ถ้า Kp มีค่ามากอาจทำให้เกิดสัญญาณเอาต์พุตเกินค่าเป้าหมาย แต่ถ้า Kp มีค่าน้อยจะทำให้ใช้เวลานานในการเข้าสู่ค่าเป้าหมาย

P:Kp*e(t)

Kp คือค่าเกณฑ์ เป็นค่าคงที่ เช่น 1.5, 1.8 หรือ 2

e(t) = Error = Setpoint - PV คือค่าผิดพลาด

เขียนเป็นอัลกอริทึมของการควบคุมแบบ P ได้ดังนี้

```
Error=Setpoint-PV;
P=Kp*Error;
```

2) **การควบคุมแบบ I หรือฟังก์ชัน I (Integral)** เป็นการอินทิเกรตสัญญาณผิดพลาดแล้วคูณกับค่า Ki หรือรวมเอาค่าผิดพลาดทั้งหมดมาคูณค่า Ki เพื่อเร่งเวลาการเข้าสู่ค่าเป้าหมายและลดค่าผิดพลาดที่เหลืออยู่ที่เกิดจากการใช้ฟังก์ชัน P

$$I:Ki*\int_0^t e(t)dt$$

Ki คือค่าเกณฑ์ เป็นค่าคงที่

Ierror = Ierror + Error คือผลรวมของค่าผิดพลาดทั้งหมดจากเวลาเริ่มต้นถึงเวลาขณะที่รันโปรแกรม (0-t)

เขียนเป็นอัลกอริทึมของการควบคุมแบบ I ได้ดังนี้

```
Error=Setpoint-PV;
Ierror=Ierror+Error;
I=Ki*Ierror;
```

3) **การควบคุมแบบ D หรือฟังก์ชัน D (Derivative)** เป็นการหาค่าความแตกต่างของสัญญาณผิดพลาดปัจจุบันกับค่าผิดพลาดก่อนหน้านั้นแล้วนำมาคูณกับค่า Kd เพื่อลดค่าผิดพลาดในระยะคงตัว

$$D:Kd*de(t)/dt$$

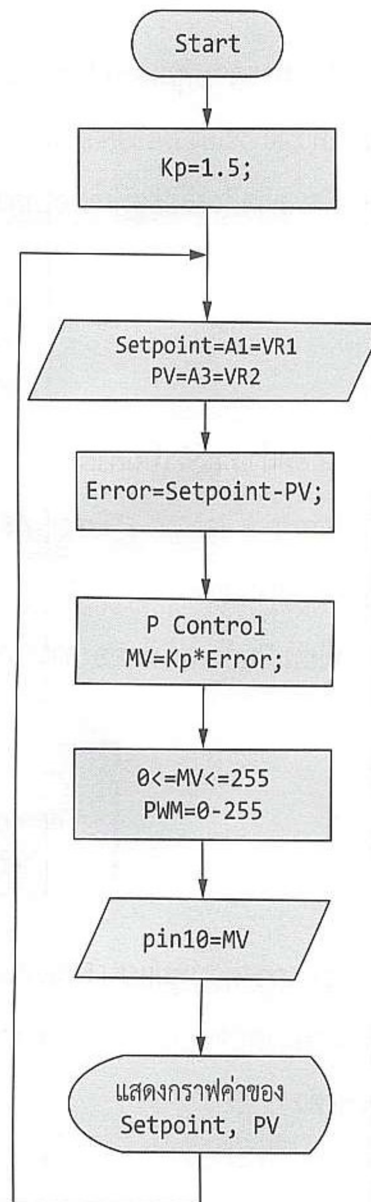
Kd คือค่าเกณฑ์ เป็นค่าคงที่ ซึ่งต้องปรับหรือตั้งค่าตามการตอบสนองของระบบ

Derror = Error - Pre_Error คือผลต่างของค่าผิดพลาดปัจจุบันกับค่าผิดพลาดก่อนหน้านั้น เขียนเป็นอัลกอริทึมของการควบคุมแบบ D ได้ดังนี้

```
Error=Setpoint-PV;
Derror=Error-Pre_Error;
D=Kd*Derror;
Pre_Error=Error;
```

ตัวอย่างที่ 9.3 โปรแกรมควบคุมตำแหน่งคานแบบ P

```
int Setpoint=0;
int Error=0;
int PV=0,MV=0;
double Kp=1.5;
void setup()
{ Serial.begin(9600);
  pinMode(10,OUTPUT);
}
void loop()
{ Setpoint=analogRead(A1);
        PV=analogRead(A3);
  Error=Setpoint-PV;
  MV=Kp*Error; // P Control
  if(MV>255) MV=255;
  if(MV<0)   MV=0;
  analogWrite(10,MV);
 // Plot Graph
 Serial.print(Setpoint);
 Serial.print("\t");
 Serial.print(PV);
 Serial.print("\t");
 Serial.print(700);
 Serial.print("\t");
 Serial.print(300);
}
```



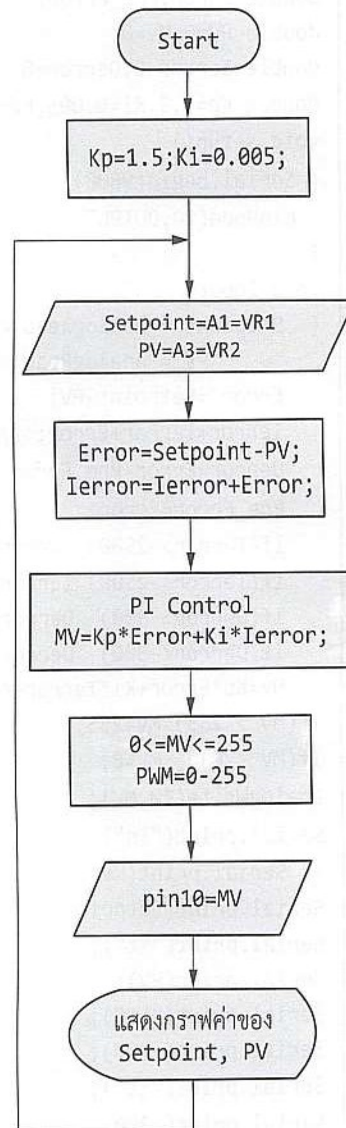
รูปที่ 9.15 โฟลว์ชาร์ตการควบคุมแบบ P

ผลการรันโปรแกรม

ไมโครคอนโทรลเลอร์จะหาค่าผิดพลาดของตำแหน่ง (Error = Setpoint - PV) จากนั้นคำนวณหาค่าเอาต์พุตด้วยการควบคุมแบบ P คือ MV = Kp*Error แล้วส่งค่าสัญญาณเอาต์พุตแบบ PWM ออกขา 10 เพื่อไปควบคุมมอเตอร์ให้หมุนไปยังตำแหน่งที่ตั้งไว้ ซึ่งอาจมีค่าผิดพลาดและแกว่งเล็กน้อย ต้องปรับค่าเกณฑ์ Kp ให้เหมาะสม คานก็จะไม่แกว่ง มีความเสถียรดีกว่าการควบคุมแบบเปิด-ปิด

ตัวอย่างที่ 9.4 โปรแกรมควบคุมตำแหน่งคานแบบ PI

```
double Setpoint,VP,Error;
double PV,MV,Ierror;
double Kp=1.5;
double Ki=0.005;
void setup()
{ Serial.begin(9600);
  pinMode(10,OUTPUT);
}
void loop()
{ Setpoint=analogRead(A1);
        PV=analogRead(A3);
   Error=Setpoint-PV;
   Ierror=Ierror+Error; // Integral Error
   if(Ierror> 2500) Ierror=2500;
   if(Ierror<-2500) Ierror=-2500;
   MV=Kp*Error+Ki*Ierror; // PI Control
   if(MV>255) MV=255;
   if(MV<0)   MV=0;
   analogWrite(10,MV); // PWM Output
  // Plot Graph
  Serial.print(“\n”);
  Serial.print(Setpoint);
  Serial.print(“\t”);
  Serial.print(PV);
  Serial.print(“\t”);
  Serial.print( 700);
  Serial.print(“\t”);
  Serial.print( 300);
}
```



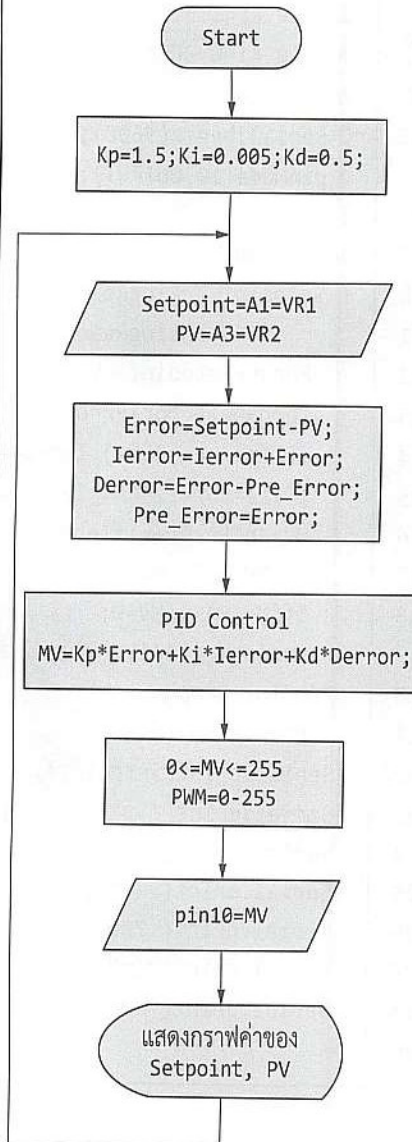
รูปที่ 9.16 โฟลว์ชาร์ตการควบคุมแบบ PI

ผลการรันโปรแกรม

ไมโครคอนโทรลเลอร์จะหาค่าผิดพลาดของตำแหน่ง (Error = Setpoint - PV) จากนั้นคำนวณหาค่าเอาต์พุตด้วยการควบคุมแบบ P และ I นำสัญญาณเอาต์พุตมารวมกันแล้วส่งสัญญาณ PWM ออกขา 10 เพื่อไปควบคุมมอเตอร์ให้หมุนไปยังตำแหน่งที่ตั้งไว้ ซึ่งอาจมีค่าผิดพลาดและแกว่งเล็กน้อย ต้องปรับค่าเกณฑ์ Kp และ Ki ให้เหมาะสม คานก็จะไม่แกว่ง มีความเสถียร ซึ่งการควบคุมแบบ PI จะทำให้ลดเวลาในการเข้าสู่ค่าตำแหน่งเป้าหมาย

ตัวอย่างที่ 9.5 โปรแกรมควบคุมตำแหน่งคานแบบ PID

```
double Setpoint=0,VP=0;
double Error,Pre_Error;
double PV=0,MV=0;
double Ierror=0,Derror=0;
double Kp=1.5,Ki=0.005,Kd=0.5;
void setup()
{ Serial.begin(9600);
  pinMode(10,OUTPUT);
}
void loop()
{  Setpoint = analogRead(A1);
         PV = analogRead(A3);
   Error =Setpoint-PV;
   Ierror=Ierror+Error; // Integrate
   Derror=Error-Pre_Error; // Derivative
   Pre_Error=Error;
   if(Ierror> 2500) Ierror=2500;
   if(Ierror<-2500) Ierror=-2500;
   if(Derror> 300)  Derror=300;
   if(Derror<-300)  Derror=-300;
   MV=Kp*Error+Ki*Ierror+Kd*Derror; // P+I+D
if(MV > 255) MV=255;
if(MV < 0)   MV=0;
analogWrite(10,MV);
Serial.print("\n");
// Serial.print(MV);
Serial.print(Setpoint);
Serial.print("\t");
Serial.print(PV);
Serial.print("\t");
Serial.print( 700);
Serial.print("\t");
Serial.print( 300);
}
```



รูปที่ 9.17 โฟลว์ชาร์ตการควบคุมแบบ PID

ผลการรันโปรแกรม

ไมโครคอนโทรลเลอร์จะหาค่าผิดพลาดของตำแหน่ง (Error = Setpoint - PV) จากนั้นคำนวณหาค่าเอาต์พุตด้วยการควบคุมแบบ PID นำสัญญาณเอาต์พุตมารวมกันแล้วส่งสัญญาณ PWM ออกขา 10 เพื่อไปควบคุมมอเตอร์ให้หมุนไปยังตำแหน่งที่ตั้งไว้ ซึ่งอาจมีค่าผิดพลาดและแกว่งเล็กน้อย ต้องปรับค่าเกณฑ์ Kp, Ki และ Kd ระบบก็จะมีความเสถียร

ตัวอย่างที่ 9.6 การควบคุมตำแหน่งคานแบบ PID โดยใช้ฟังก์ชัน

```
#include <PID_v1.h>
double Setpoint,PV,MV,feedback;
double Kp=1.5,Ki=0.05,Kd=0.5;
PID myPID(&PV,&MV,&Setpoint,Kp,Ki,Kd,DIRECT);
void setup()
{
  pinMode(10,OUTPUT);
  myPID.SetMode(AUTOMATIC);
  myPID.SetSampleTime(50);
  myPID.SetOutputLimits(0,255);
  Serial.begin(9600);
}
void loop()
{ Setpoint=analogRead(A1);
  feedback=analogRead(A3);
  myPID.Compute();
  analogWrite(10,MV);
  Serial.print(“\n”);
  Serial.print(Setpoint);
  Serial.print(“\t”);
  Serial.print(feedback);
  Serial.print(“\t”);
 }
```

ผลการรันโปรแกรม

ไมโครคอนโทรลเลอร์จะหาค่าผิดพลาดของตำแหน่ง (Error = Setpoint - PV) จากนั้นคำนวณหาค่าเอาต์พุตด้วยการควบคุมแบบ PID ด้วยฟังก์ชัน myPID.Compute(); แล้วนำสัญญาณเอาต์พุตที่คำนวณได้ในตัวแปร MV ส่งสัญญาณออกขา 10 เพื่อไปควบคุมมอเตอร์ให้หมุนไปยังตำแหน่งที่ตั้งไว้ การควบคุมตำแหน่งหากมีค่าผิดพลาดหรือแกว่งเล็กน้อย ต้องไปปรับค่าเกณฑ์ Kp, Ki และ Kd ระบบก็จะมีความเสถียร

จากตัวอย่างที่ 9.6 การควบคุมตำแหน่งคานโดยใช้ฟังก์ชัน PID มีหน้าที่การทำงานดังนี้

```
#include <PID_v1.h>
```

เรียกใช้งานไลบรารี PID_v1.h เพื่อใช้ฟังก์ชันต่าง ๆ ของ PID ได้

```
PID myPID(&PV,&MV,&Setpoint,Kp,Ki,Kd,DIRECT);
```

เปิดใช้งานฟังก์ชัน PID และกำหนดค่าตัวแปรต่าง ๆ

```
myPID.SetMode(AUTOMATIC);
```

กำหนดโหมดการทำงานของฟังก์ชัน PID แบบอัตโนมัติ

```
myPID.SetSampleTime(50);
```

กำหนดจังหวะเวลาการทำงาน

```
myPID.SetOutputLimits(0,255);
```

กำหนดค่าเอาต์พุตที่ส่งออกให้อยู่ระหว่าง 0 ถึง 255

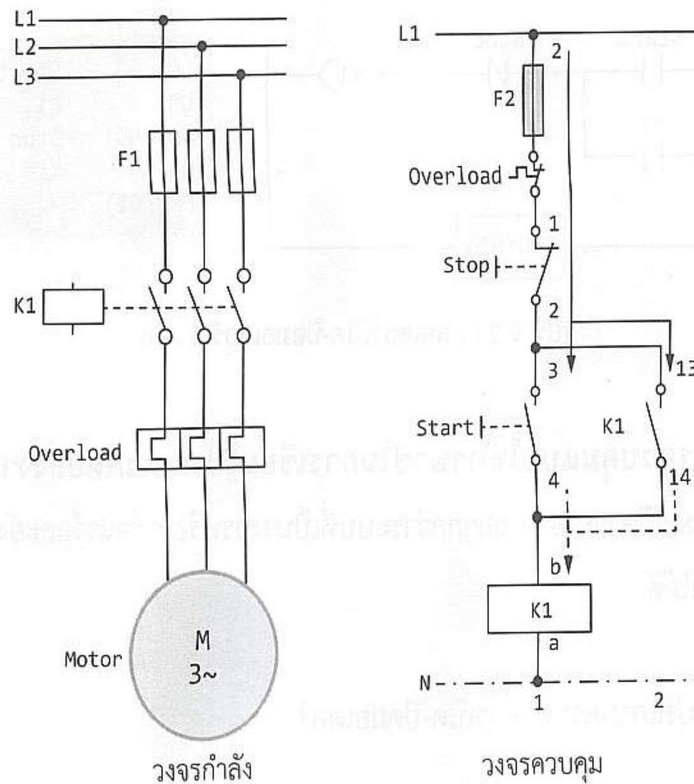
```
myPID.Compute();
```

คำนวณหาค่าสัญญาณเอาต์พุตตามวิธีการของ PID โดยสัญญาณเอาต์พุตที่ได้จะมีค่า 0 ถึง 255

9.8 ความสัมพันธ์ระหว่างวงจรควบคุมกับโปรแกรม

การควบคุมในยุคเริ่มต้นจะเป็นการควบคุมโดยใช้คอนแทกเตอร์หรือวงจรไฟฟ้าเป็นตัวควบคุม เมื่อเทคโนโลยีก้าวหน้า ได้พัฒนาเป็นระบบควบคุมด้วยไมโครคอนโทรลเลอร์ คอมพิวเตอร์และโปรแกรม ตัวอย่างเช่น การควบคุมมอเตอร์ 3 เฟส เพื่อเชื่อมโยงให้เห็นถึงความสัมพันธ์ระหว่างวงจรไฟฟ้าซึ่งเป็น ฮาร์ดแวร์ วงจรดิจิทัล ภาษาแลดเดอร์ และการเขียนโปรแกรมภาษาซีซึ่งเป็นซอฟต์แวร์

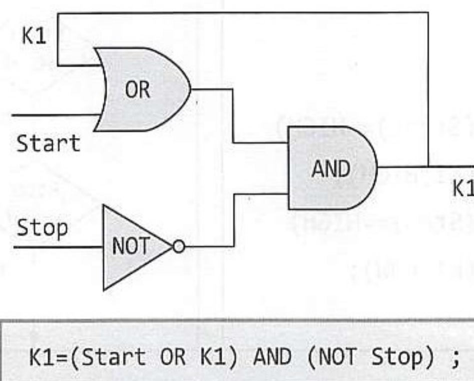
1) **การควบคุมมอเตอร์โดยใช้คอนแทกเตอร์** เป็นการควบคุมที่ใช้อุปกรณ์ที่เป็นฮาร์ดแวร์ทั้งหมด



รูปที่ 9.18 วงจรเปิด-ปิดมอเตอร์ 3 เฟส

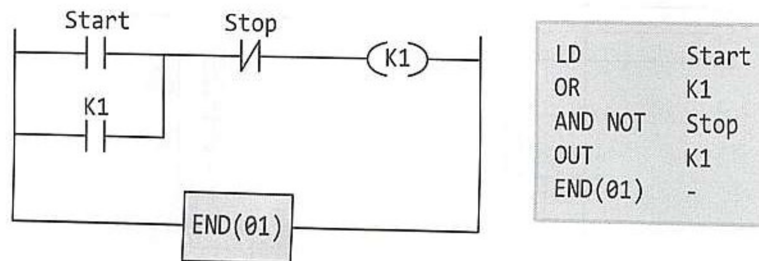
การทำงานของวงจรเมื่อกดสวิตช์ Start คอนแทกเตอร์ K1 จะทำงาน ส่งผลให้มอเตอร์หมุน และเมื่อกดสวิตช์ Stop มอเตอร์จะหยุดทำงาน

2) **วงจรควบคุมแบบดิจิทัล** เป็นฮาร์ดแวร์หรือวงจรที่ใช้กับสัญญาณดิจิทัล โดยใช้วงจรเกตมาต่อกันเพื่อควบคุม มีลักษณะการทำงานเหมือนกับวงจรไฟฟ้าแต่ใช้ได้กับสัญญาณดิจิทัล



รูปที่ 9.19 วงจรดิจิทัลการเปิด-ปิดมอเตอร์

3) การควบคุมแบบใช้โปรแกรมแลดเดอร์ (Ladder) การควบคุมที่ผสมระหว่างฮาร์ดแวร์และซอฟต์แวร์คือโปรแกรมเมเบิลคอนโทรลเลอร์และโปรแกรมแลดเดอร์

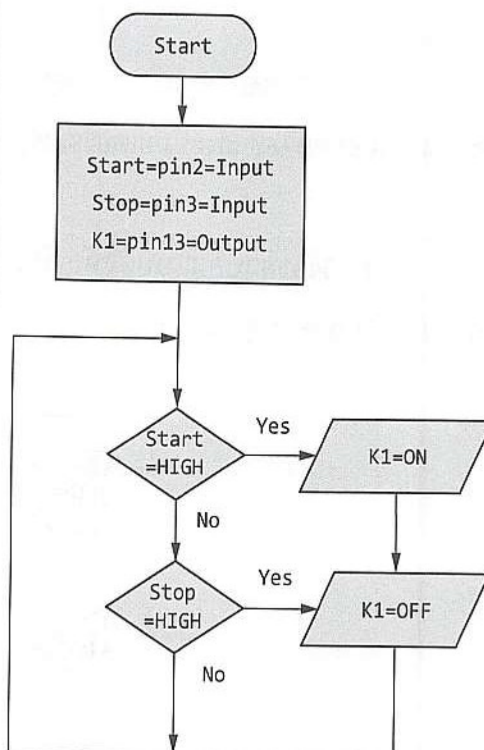


รูปที่ 9.20 แลดเดอร์เปิด-ปิดมอเตอร์ 3 เฟส

4) การควบคุมแบบใช้ภาษาซีในการเขียนโปรแกรมโดยสั่งงานผ่านไมโครคอนโทรลเลอร์ ซึ่งจะมีความยืดหยุ่นมากกว่าระบบที่เป็นวงจรหรือฮาร์ดแวร์และยังสามารถทำงานในระบบที่ซับซ้อนได้ดี

ตัวอย่างที่ 9.7 โปรแกรมควบคุมการเปิด-ปิดมอเตอร์

```
int Start=2;
int Stop=3;
int K1=13;
void setup()
{
  Serial.begin(9600);
  pinMode(Start,INPUT);
  pinMode(Stop,INPUT);
  pinMode(K1,OUTPUT);
}
void loop()
{
 if(digitalRead(Start)==HIGH)
   digitalWrite(K1,HIGH);
 if(digitalRead(Stop)==HIGH)
   digitalWrite(K1,LOW);
}
```



รูปที่ 9.21 โฟลว์ชาร์ตการควบคุมมอเตอร์

ผลการรันโปรแกรม

เมื่อกดสวิตช์ Start ที่ต่อกับขา 2 จะทำให้ K1 เอาต์พุตขา 13 ทำงาน และเมื่อกดสวิตช์ Stop ที่ขา 3 จะทำให้ K1 เอาต์พุตขา 13 หยุดทำงาน

9.9 สรุป

ระบบควบคุม หมายถึง ระบบการบังคับ ควบคุม สั่งการเพื่อควบคุมกระบวนการให้ทำงานตามวัตถุประสงค์ที่ต้องการ ซึ่งระบบควบคุมที่ดีควรมีเสถียรภาพ สามารถควบคุมสัญญาณเอาต์พุตให้เป็นไปตามค่าที่ตั้งไว้ มีการตอบสนองทันเวลา สัญญาณเอาต์พุตไม่แกว่งและมีค่าผิดพลาดน้อย

ระบบควบคุมแบบเปิด เป็นระบบควบคุมแบบทิศทางเดียว คือส่งสัญญาณอินพุตไปยังตัวควบคุมเพื่อควบคุมกระบวนการ เป็นระบบควบคุมที่ไม่มีการป้อนกลับของสัญญาณเอาต์พุตเพื่อมาเปรียบเทียบหรือตรวจสอบความถูกต้องกับสัญญาณอินพุต ข้อดีของระบบควบคุมแบบเปิด คือเป็นระบบควบคุมที่ง่าย ไม่ซับซ้อน ทนทาน และมีราคาถูก

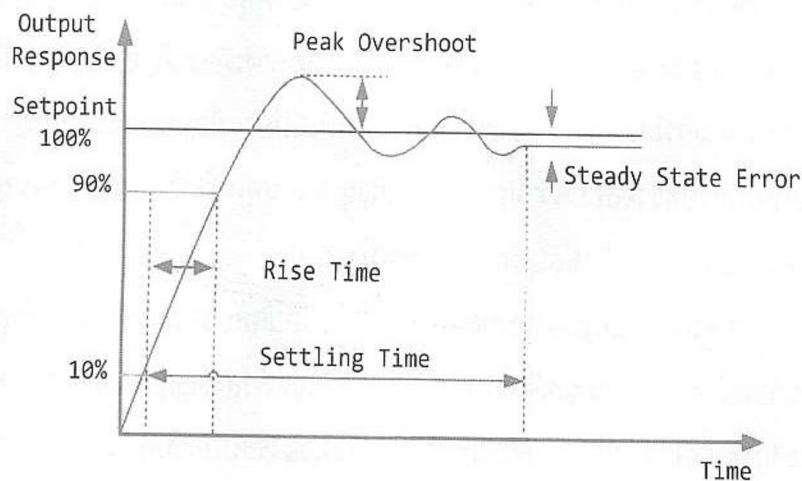
ระบบควบคุมแบบปิด เป็นระบบควบคุมที่มีการป้อนกลับของสัญญาณเอาต์พุตเพื่อนำมาเปรียบเทียบหรือตรวจสอบความถูกต้องกับสัญญาณอินพุต แล้วส่งสัญญาณให้ตัวควบคุมเพื่อปรับการทำงานของกระบวนการให้ถูกต้องตามค่าที่ตั้งไว้ ข้อดีของระบบควบคุมแบบปิด คือเป็นระบบที่ควบคุมให้สัญญาณเอาต์พุตมีความถูกต้องได้มากกว่าระบบควบคุมแบบเปิด แต่ก็มีการทำงานที่ซับซ้อนและมีราคาแพง

ระบบควบคุมแบบเปิด-ปิด เป็นระบบควบคุมให้เอาต์พุตทำงานเพียง 2 สภาวะเท่านั้น คือ เปิดและปิด หรือ ON และ OFF เป็นการควบคุมแบบ 2 ตำแหน่งอย่างง่าย มีราคาถูก สัญญาณเอาต์พุตมีการแกว่ง เกิดการเปลี่ยนแปลงไม่คงที่

ระบบควบคุมแบบ PID เป็นระบบควบคุมแบบป้อนกลับชนิดหนึ่งที่ได้รับความนิยมและมีการใช้งานทั่วไป เช่น ใช้ในการควบคุมอุณหภูมิ ใช้ควบคุมตำแหน่งและความเร็ว หลักการทำงานของระบบ PID จะนำค่าผิดพลาด e(t) มาผ่านฟังก์ชัน P ซึ่งจะทำหน้าที่ปรับเกณฑ์การขยายของสัญญาณ ฟังก์ชัน I ทำหน้าที่อินทิเกรต และฟังก์ชัน D ทำหน้าที่อนุพันธ์ แล้วนำสัญญาณทั้งหมดมารวมกันเพื่อส่งออกไปควบคุมอุปกรณ์หรือกระบวนการ โดยระบบควบคุมจะพยายามลดค่าผิดพลาดให้เหลือน้อยที่สุดตามการปรับค่าตัวแปรของ PID เพื่อให้เอาต์พุตของระบบได้ตามค่าที่กำหนดไว้

คำถามท้ายบทที่ 9

1. จงเขียนบล็อกไดอะแกรมของระบบควบคุมแบบพื้นฐาน
2. จงยกตัวอย่างของระบบควบคุมแบบป้อนกลับพร้อมเขียนบล็อกไดอะแกรมและอธิบายการทำงาน
3. จงอธิบายผลตอบสนองเอาต์พุตของระบบควบคุม โดยอธิบายความหมายของคำต่อไปนี้ Setpoint, Rise Time, Settling Time, Peak Overshoot



4. จงอธิบายการทำงานของระบบควบคุมแบบเปิด-ปิดพร้อมยกตัวอย่างประกอบ
5. จงอธิบายการทำงานของระบบควบคุมแบบ P และเขียนอัลกอริทึมหาค่าเอาต์พุตของการควบคุมแบบ P
6. จงอธิบายการทำงานของระบบควบคุมแบบ I และเขียนอัลกอริทึมหาค่าเอาต์พุตของการควบคุมแบบ I
7. จงอธิบายการทำงานของระบบควบคุมแบบ D และเขียนอัลกอริทึมหาค่าเอาต์พุตของการควบคุมแบบ D

10

หุ่นยนต์และแขนกล

คำว่า Robot แปลว่า หุ่นยนต์ มาจากคำว่า Robota ในภาษาเช็ก แปลว่า การทำงานเสมือนทาส ในปี ค.ศ. 1921 คาเรล คาเปก (Karel Capek) ได้ประพันธ์บทละครเวทีเรื่อง "Rossum's Universal Robots" ที่มีเนื้อหาเกี่ยวกับการสร้างหุ่นยนต์ขึ้นมาเพื่อรับใช้มนุษย์เสมือนทาส จนกระทั่งหุ่นยนต์มีวิวัฒนาการทางความคิดเสมือนมนุษย์ทำให้เกิดการต่อต้านจากหุ่นยนต์จากนิยายในอดีตสู่โลกที่เป็นความจริงในปัจจุบัน หุ่นยนต์ได้ถูกวิจัยและพัฒนาโดยหน่วยงานและสถาบันต่าง ๆ ให้มีความสามารถเพิ่มขึ้น จนทำให้หุ่นยนต์เริ่มเข้ามามีบทบาทในชีวิตประจำวันของมนุษย์เรื่อยมา เทคโนโลยีที่ได้รับการพัฒนาอย่างต่อเนื่องในปัจจุบันทำให้ความสามารถของหุ่นยนต์พัฒนาขึ้นอย่างมาก สามารถเคลื่อนที่ได้เร็ว คิดและตัดสินใจได้ ประมวลผลสัญญาณภาพได้ แสดงความรู้สึกได้ และทำงานบางอย่างที่มนุษย์ไม่สามารถทำได้

10.1 หุ่นยนต์คืออะไร ?

หุ่นยนต์ คือ เครื่องจักรกลอัตโนมัติ

ความหมายของหุ่นยนต์โดยสถาบันหุ่นยนต์แห่งสหรัฐอเมริกา (The Robotics Institute of America: RIA) ได้ให้ความหมายไว้ดังนี้

"A robot is a reprogrammable, multifunctional manipulator designed to move materials, parts, tools or specialized devices through various programmed motions for the performance of a variety of tasks."

"หุ่นยนต์ คือ เครื่องจักรกลที่สามารถทำการโปรแกรมซ้ำได้ สามารถทำงานได้หลาย ๆ หน้าที่ ได้รับการออกแบบมาเพื่อให้สามารถหยิบ จับ ย้าย วัสดุ อุปกรณ์ เครื่องมือหรืออุปกรณ์พิเศษต่าง ๆ โดยการตั้งโปรแกรมควบคุมการเคลื่อนที่ให้ทำงานได้ตามต้องการ"

หุ่นยนต์มีหลากหลายประเภทขึ้นอยู่กับวัตถุประสงค์ในการออกแบบและการใช้งาน หากจำแนกหุ่นยนต์ตามลักษณะการเคลื่อนที่ สามารถแบ่งได้เป็น 2 ประเภทใหญ่ ๆ คือ

1) **หุ่นยนต์ชนิดติดตั้งอยู่กับที่ (Fixed Robot)** หุ่นยนต์ประเภทนี้มีลักษณะเป็นแขนกล ซึ่งสามารถเคลื่อนไหวได้เฉพาะข้อต่อในพื้นที่จำกัด ส่วนมากใช้ในโรงงานอุตสาหกรรม

2) **หุ่นยนต์ชนิดเคลื่อนที่ได้ (Mobile Robot)** หุ่นยนต์ประเภทนี้สามารถเคลื่อนที่ได้ด้วยการใช้ล้อ ขา ปีก หรือการเคลื่อนที่ในรูปแบบอื่น ๆ

นอกจากนี้ยังมีหุ่นยนต์อีกหลายประเภทที่หลายหน่วยงานเป็นผู้พัฒนาขึ้น ตัวอย่างเช่น

- หุ่นยนต์ฮิวแมนนอยด์ (Humanoid Robot) หุ่นยนต์ที่มีลักษณะเหมือนกับมนุษย์
- แอนดรอยด์ (Android) หุ่นยนต์ที่สามารถแสดงออกเหมือนมนุษย์
- ไซบอร์ก (Cyborg) เป็นหุ่นยนต์ที่ผสมผสานกันระหว่างสิ่งมีชีวิตกับเครื่องจักร หรือครึ่งคนครึ่งหุ่นยนต์
- นาโนบอท (Nanobot) หุ่นยนต์ที่มีขนาดเล็กมาก ขนาดประมาณ 0.5-3 ไมครอน

ความสามารถในการทำงานของหุ่นยนต์สามารถจำแนกได้ 6 ระดับ ตามเกณฑ์มาตรฐานของสมาคมหุ่นยนต์อุตสาหกรรมแห่งญี่ปุ่น (Japan Industrial Robot Association: JIRA) ดังนี้

ระดับที่ 1 กลไกที่ถูกควบคุมด้วยมนุษย์หรือหุ่นยนต์ที่ควบคุมด้วยมือ (Manual-handling Device) เช่น หุ่นยนต์ม้าแข่งและหุ่นยนต์บังคับด้วยมือ

ระดับที่ 2 หุ่นยนต์ที่ทำงานตามการโปรแกรมหรือขั้นตอนที่กำหนดไว้ แต่ไม่สามารถปรับเปลี่ยนการทำงานได้ (Fixed-sequence Robot) เช่น หุ่นยนต์เดินตามเส้นที่ไม่ใช้ไมโคร-คอนโทรลเลอร์เป็นตัวควบคุม

ระดับที่ 3 หุ่นยนต์ที่ทำงานตามการโปรแกรมและสามารถปรับเปลี่ยนแก้ไขโปรแกรมได้ (Variable-sequence Robot) เช่น หุ่นยนต์ที่ควบคุมและประมวลผลข้อมูลด้วยไมโครโปรเซสเซอร์หรือไมโครคอนโทรลเลอร์

ระดับที่ 4 หุ่นยนต์ที่มีผู้ควบคุมเป็นผู้สอนงานให้ โดยหุ่นยนต์จะทำงานตามหน่วยความจำที่บันทึกไว้ (Playback Robot)

ระดับที่ 5 หุ่นยนต์ที่ทำงานตามการบันทึกโปรแกรมข้อมูลเชิงตัวเลข เพื่อกำหนดตำแหน่งการเคลื่อนที่ของหุ่นยนต์ให้ทำงานอัตโนมัติตามข้อมูลตัวเลขที่โปรแกรม (Numerical Control Robot)

ระดับที่ 6 หุ่นยนต์ที่มีความฉลาดสามารถเรียนรู้จากสภาพแวดล้อมและตัดสินใจทำงานได้ด้วยตัวเอง (Intelligent Robot)

สำหรับสถาบันหุ่นยนต์แห่งสหรัฐอเมริกา กำหนดให้หุ่นยนต์ต้องมีความสามารถในระดับที่ 3-6 เท่านั้นจึงจะเรียกได้ว่าหุ่นยนต์

10.2 ส่วนประกอบของหุ่นยนต์

ในการสร้างหุ่นยนต์ต้องใช้ศาสตร์หลายสาขามาบูรณาการรวมกัน เช่น วิศวกรรมเครื่องกล วิศวกรรมไฟฟ้า และวิศวกรรมคอมพิวเตอร์ โดยองค์ประกอบของหุ่นยนต์จะประกอบไปด้วยโครงสร้างของหุ่นยนต์ กลไก ข้อต่อต่าง ๆ รวมถึงวงจรไฟฟ้า วงจรอิเล็กทรอนิกส์ เซนเซอร์ ไมโครคอนโทรลเลอร์ และโปรแกรมหรือซอฟต์แวร์ ส่วนประกอบของหุ่นยนต์แบ่งออกได้เป็น 5 ส่วนดังนี้

ส่วนที่ 1 โครงสร้างและกลไก

เป็นส่วนประกอบของโครงสร้างทางกายภาพของหุ่นยนต์ ซึ่งประกอบไปด้วย แขน ขา ข้อต่อ ข้อเหวี่ยง ล้อ สายพาน มือจับ เฟือง และกลไกต่าง ๆ ที่ประกอบขึ้นเป็นตัวหุ่นยนต์ หากเปรียบกับมนุษย์ก็คือ แขน ขา นิ้วมือ กระดูก และข้อต่อต่าง ๆ ที่รวมกันเป็นร่างกาย

ส่วนที่ 2 ตัวขับเคลื่อน

ตัวขับเคลื่อนคือส่วนที่ทำให้หุ่นยนต์เกิดการเคลื่อนที่ ส่วนใหญ่เป็นการเคลื่อนที่โดยการหมุนเป็นวงกลม เช่น ดีซีมอเตอร์ เซอร์โวมอเตอร์ สเต็ปเปอร์มอเตอร์ สำหรับตัวขับเคลื่อนที่ทำให้เกิดการเคลื่อนที่แบบเส้นตรง เช่น วาล์วไฟฟ้า กระบอกนิวแมติกและไฮดรอลิก หากเปรียบกับร่างกายมนุษย์ก็คือ กล้ามเนื้อและเส้นเอ็นต่าง ๆ ที่ควบคุมการเคลื่อนที่ของร่างกาย

ส่วนที่ 3 เซนเซอร์

เซนเซอร์เป็นอุปกรณ์ที่ใช้ตรวจวัดสัญญาณ ข้อมูล หรือสภาวะแวดล้อมเพื่อส่งให้วงจรควบคุมนำไปประมวลผล เช่น เซนเซอร์ตรวจวัดสี เสียง แสง อุณหภูมิ ตำแหน่งการเคลื่อนที่ และภาพเคลื่อนไหว หากเปรียบกับร่างกายมนุษย์ก็คือ หู ตา จมูก ลิ้น และประสาทสัมผัสต่าง ๆ ที่ทำให้เกิดการรับรู้

ส่วนที่ 4 วงจรควบคุมและไมโครคอนโทรลเลอร์

วงจรควบคุมเป็นส่วนที่ควบคุมการทำงานของหุ่นยนต์ ซึ่งประกอบไปด้วยวงจรไฟฟ้า วงจรอิเล็กทรอนิกส์ วงจรขยายสัญญาณ และไมโครคอนโทรลเลอร์

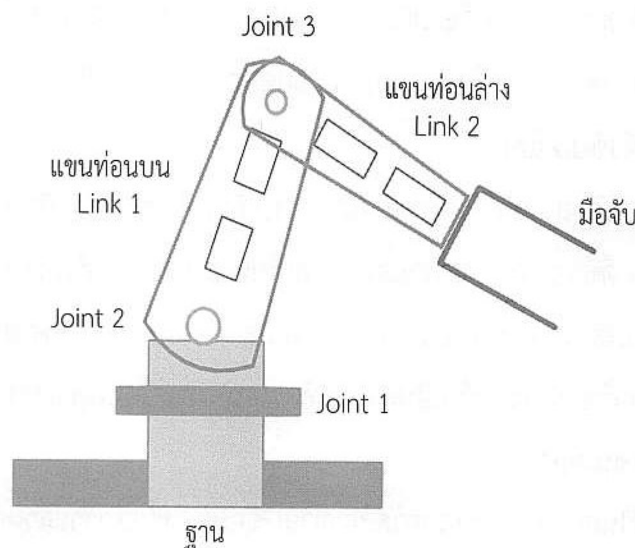
ส่วนที่ 5 โปรแกรมหรือซอฟต์แวร์

โปรแกรมหรือซอฟต์แวร์ทำหน้าที่เสมือนเป็นสมองของหุ่นยนต์ ซึ่งทำหน้าที่ คิด คำนวณ และประมวลผลตามการโปรแกรม โดยส่วนใหญ่จะใช้โปรแกรมภาษาแอสเซมบลี ภาษาเบสิก ภาษาซี หรือภาษาภาพ (Graphics Programming)

10.3 แขนกล

แขนกล (Robot Arm) เป็นหุ่นยนต์ที่มีฐานยึดติดกับที่ มักถูกนำมาใช้ในอุตสาหกรรมการผลิตเพื่อใช้แทนแรงงานมนุษย์ในงานที่มีลักษณะซ้ำ ๆ งานอันตราย งานหนัก งานที่น่าเบื่อ หรืองานที่ต้องการความเที่ยงตรงแม่นยำสูง แขนกลอุตสาหกรรมที่ใช้โดยทั่วไป ได้แก่ แขนกลในโรงงานประกอบรถยนต์ แขนกลสำหรับงานเชื่อมและงานประกอบ

โครงสร้างของแขนกลประกอบด้วย ฐานของหุ่นยนต์ ชิ้นส่วนที่เป็นแขนกลหรือส่วนเชื่อมต่อ (Link) ข้อต่อหรือจุดหมุน (Joint) และปลายของแขนกล เช่น มือจับ หัวเจาะ หัวเชื่อม หรือหัวพ่นสี



รูปที่ 10.1 โครงสร้างของแขนกล

10.4 ชนิดของแขนกล

การแบ่งชนิดของแขนกลจะแบ่งตามลักษณะรูปทรงของพื้นที่ในการทำงาน ซึ่งแขนกลอุตสาหกรรมจะเป็นแขนกล 3 แกนขึ้นไป เพื่อให้ครอบคลุมพื้นที่ทำงานซึ่งมีลักษณะแตกต่างกันตามสภาพงานจริง สามารถแบ่งชนิดของแขนกลได้ดังนี้

1) แขนกลชนิด Cartesian

แขนกลชนิดนี้จะเคลื่อนที่เป็นเส้นตรง (Prismatic) ทั้ง 3 แกน ข้อดีของแขนกลชนิดนี้คือ ควบคุมตำแหน่งได้ง่ายเนื่องจากทุกแกนเคลื่อนที่เป็นเส้นตรง เช่น ลักษณะการทำงานของพล็อตเตอร์และเครื่องพิมพ์สามมิติ

2) แขนกลชนิด Cylindrical

แขนกลชนิดนี้มีพื้นที่การทำงานเป็นรูปทรงกระบอก เคลื่อนที่แบบหมุน (Revolute) และแบบเส้นตรงผสมกัน มีข้อดีคือการควบคุมตำแหน่งไม่ซับซ้อนและสามารถทำงานในพื้นที่แคบ ๆ หรือลักษณะที่เป็นโพรงได้

3) แขนกลชนิด Spherical

แขนกลชนิดนี้มีพื้นที่การทำงานเป็นรูปทรงกลม เคลื่อนที่ในแนวเส้นตรงและหมุนผสมกัน การควบคุมตำแหน่งและการเคลื่อนที่มีความซับซ้อนขึ้น มีข้อดีคือสามารถหมุนในมุมต่าง ๆ ได้

4) แขนกลชนิด SCARA

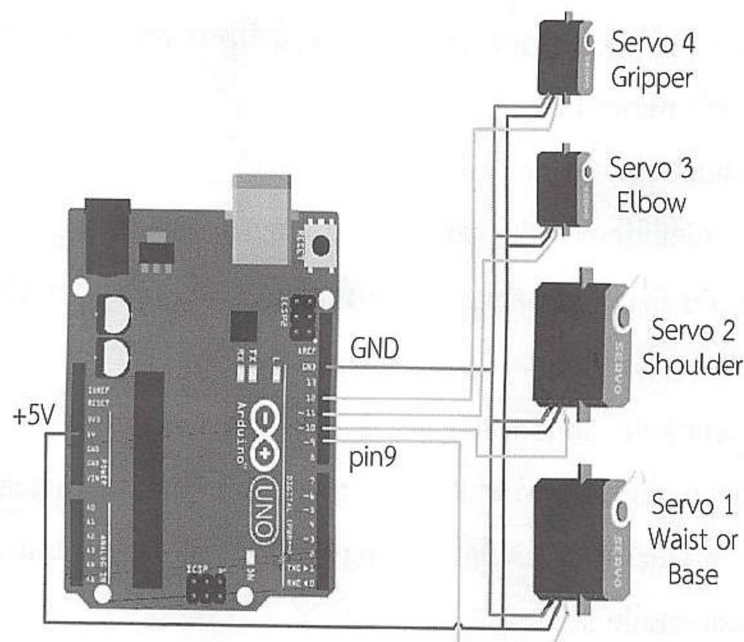
แขนกลชนิด SCARA (Selective Compliance Assembly Robot Arm) การเคลื่อนที่ส่วนฐานและข้อศอกจะเป็นแบบหมุน การเคลื่อนที่ในแนวตั้งจะเป็นการเคลื่อนที่ขึ้นลงในแนวแกน แขนกลชนิด SCARA สามารถเคลื่อนที่ได้เร็วในแนวระนาบและมีความแม่นยำสูง

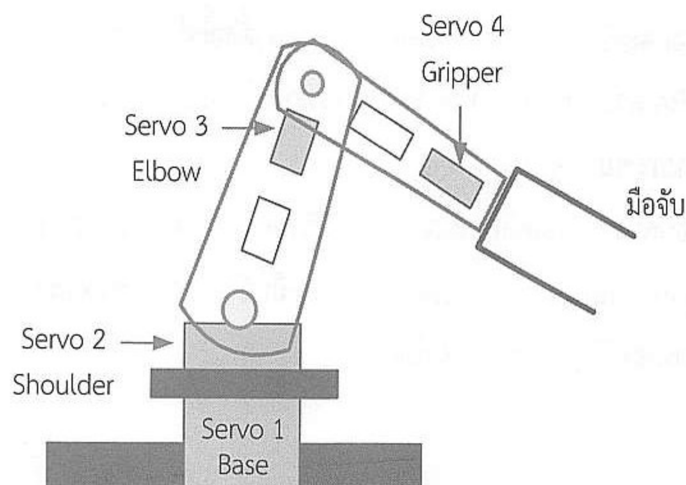
5) แขนกลชนิด Articulated

แขนกลชนิดนี้ทุกแกนเคลื่อนที่แบบหมุน มีลักษณะการเคลื่อนที่คล้ายกับแขนคน ประกอบด้วยช่วงเอวหรือฐาน แขนท่อนบน แขนท่อนล่าง และมือจับ แขนกลชนิดนี้มีระบบพิกัดที่ซับซ้อนทำให้ควบคุมการเคลื่อนที่แบบเส้นตรงทำได้ยาก

10.5 วงจรควบคุมแขนกล

วงจรควบคุมแขนกลที่จะศึกษาในหัวข้อนี้ จะใช้ไมโครคอนโทรลเลอร์ Arduino UNO เป็นตัวควบคุมการทำงานของเซอร์โวมอเตอร์ทั้ง 4 ตัวเพื่อไปขับเคลื่อนกลไกการทำงานของแขนกล โดยเซอร์โวมอเตอร์ตัวที่ 1 ทำหน้าที่หมุนฐาน เซอร์โวมอเตอร์ตัวที่ 2 ทำหน้าที่หมุนหัวไหล่ เซอร์โวมอเตอร์ตัวที่ 3 ทำหน้าที่หมุนข้อศอก และเซอร์โวมอเตอร์ตัวที่ 4 ทำหน้าที่จับหรือปล่อยชิ้นงาน





รูปที่ 10.2 วงจรควบคุมแขนกลและตำแหน่งของเซอร์โวมอเตอร์

10.6 การประกอบแขนกล

ในหัวข้อนี้จะศึกษาแขนกลที่มี 3 แกนชนิด Articulated เนื่องจากมีขนาดเล็ก เหมาะสำหรับการเรียนรู้และฝึกการเขียนโปรแกรม โดยโครงสร้างของแขนกลทำจากพลาสติกขนาดเล็ก 4 ชิ้น ใช้เซอร์โวมอเตอร์ 4 ตัว น้ำหนักเบา มีโครงสร้างที่ไม่ซับซ้อน สามารถออกแบบและใช้เครื่องพิมพ์ 3 มิติในการทำชิ้นงานได้ การประกอบแขนกลสามารถประกอบตามแบบจนเสร็จแล้วจึงทดสอบการทำงาน หรืออาจจะประกอบทีละส่วนแล้วทดสอบการทำงานทีละขั้นตอนเพื่อป้องกันการทำงานผิดพลาดในการประกอบซึ่งอาจทำให้แขนกลเสียหายได้ ขั้นตอนการประกอบแขนกลมีดังนี้

1) ประกอบฐานวงกลมเล็กเข้ากับเซอร์โวมอเตอร์ 1 เพื่อทำหน้าที่เป็นฐานหมุนซ้ายขวา



รูปที่ 10.3 ประกอบฐานวงกลมด้านบนเข้ากับเซอร์โวมอเตอร์ 1

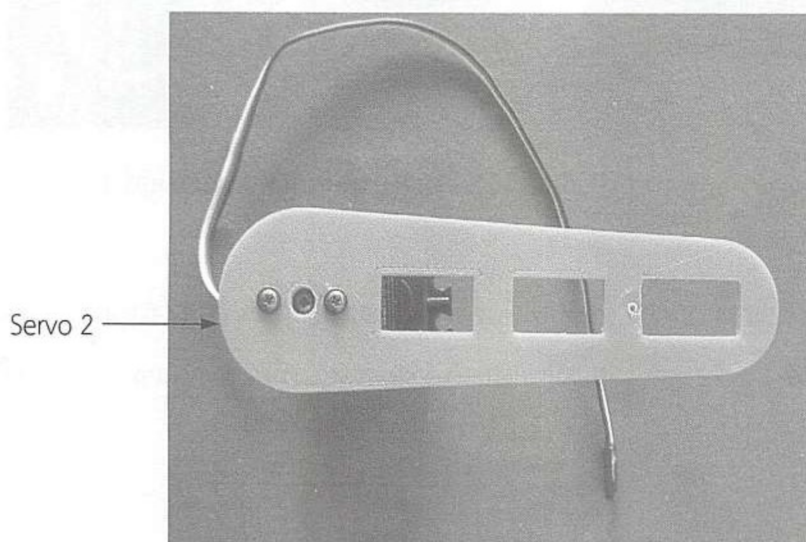
ในการประกอบฐานด้านบนเพื่อทำหน้าที่หมุนซ้ายขวา ต้องตั้งให้ตรงตำแหน่ง จากนั้นทดสอบการทำงานของฐานโดยใช้มือหมุน 0-180 องศา เมื่อแขนกลสามารถหมุนได้คล่องแล้วจึงทำขั้นตอนต่อไป

2) ใช้ปืนกาวประกอบฐานด้านล่างหรือแผ่นวงกลมใหญ่เข้ากับเซอร์โวมอเตอร์ 1



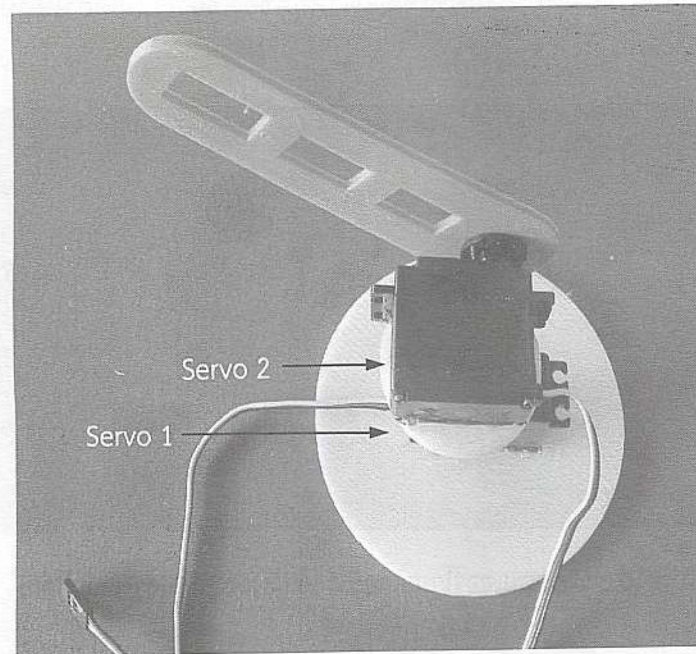
รูปที่ 10.4 ประกอบฐานวงกลมด้านล่างเข้ากับเซอร์โวมอเตอร์ 1

3) ประกอบแขนท่อนบน (แขนใหญ่) เข้ากับเซอร์โวมอเตอร์ 2 ซึ่งทำหน้าที่เสมือนหัวไหล่ของมนุษย์ เมื่อประกอบเสร็จแล้วให้ทดสอบการทำงานของหัวไหล่โดยใช้มือหมุนขึ้นลง



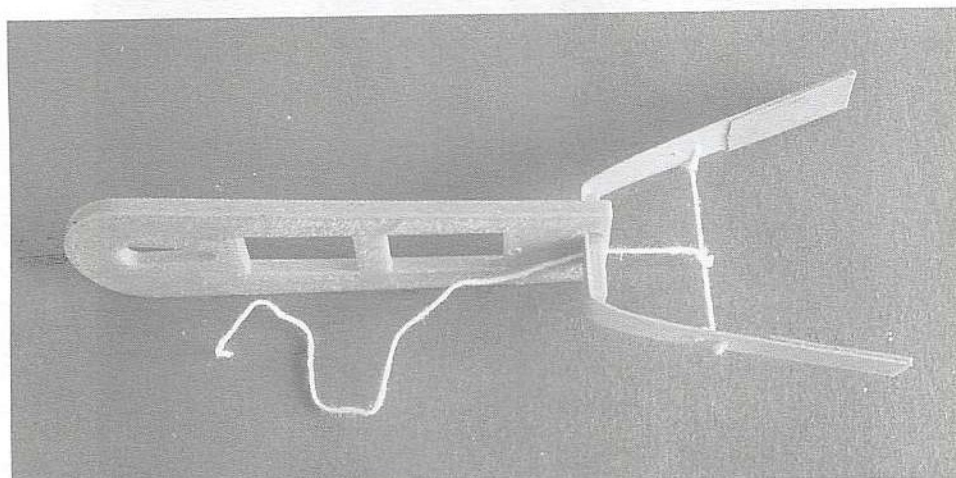
รูปที่ 10.5 ประกอบแขนท่อนบนเข้ากับเซอร์โวมอเตอร์ 2

4) ใช้ปืนกาวประกอบเซอร์โวมอเตอร์ 2 ติดกับฐานวงกลมด้านบน



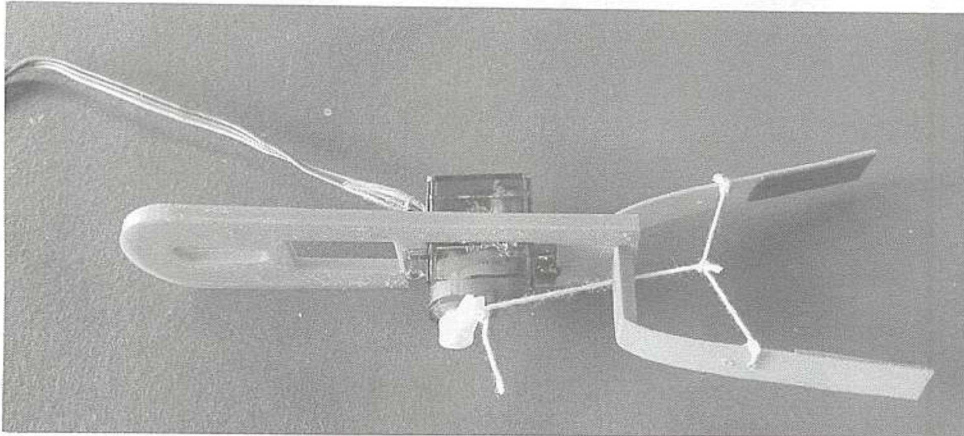
รูปที่ 10.6 ประกอบแขนท่อนบนเข้ากับฐานแขนกล

5) ประกอบมือจับเข้ากับแขนท่อนล่างดังรูปที่ 10.7



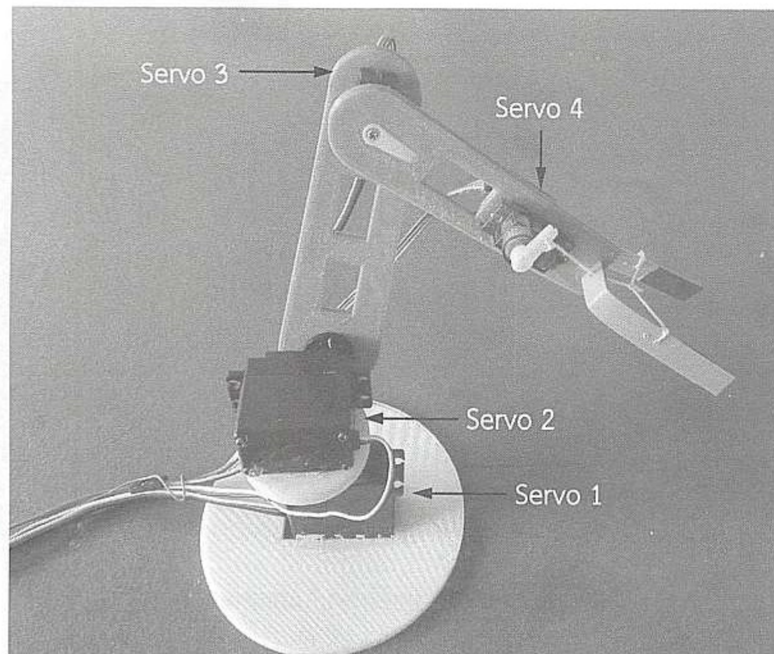
รูปที่ 10.7 ประกอบมือจับเข้ากับแขนท่อนล่าง

6) ประกอบเซอร์โวมอเตอร์ 4 เข้ากับแขนท่อนล่างและมือจับ ซึ่งส่วนนี้ทำหน้าที่เสมือนนิ้วมือของมนุษย์ เมื่อประกอบเสร็จแล้วให้ทดสอบการทำงานของมือจับว่าสามารถจับสิ่งของได้หรือไม่ หากไม่สามารถจับสิ่งของได้ให้ปรับตำแหน่งการหมุนของเซอร์โวมอเตอร์ให้เหมาะสม



รูปที่ 10.8 ประกอบเซอร์โวมอเตอร์ 4 เข้ากับแขนท่อนล่างและมือจับ

7) ประกอบแขนท่อนล่างเข้ากับเซอร์โวมอเตอร์ 3 และแขนท่อนบน ก็จะได้แขนกลเพื่อทำการทดลอง



รูปที่ 10.9 แขนกลที่ประกอบเสร็จแล้ว

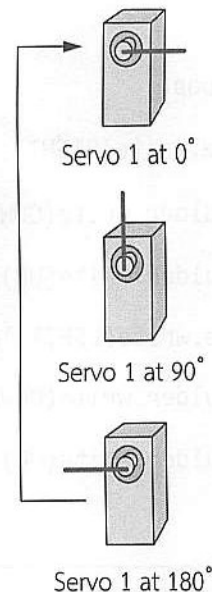
***หมายเหตุ :** สามารถสอบถามข้อมูลเพิ่มเติมจากผู้เขียนได้ตามข้อมูลติดต่อท้ายเล่ม*

10.7 การเขียนโปรแกรมควบคุมแขนกล

ในหัวข้อนี้จะเขียนโปรแกรมควบคุมแขนกล ด้วยการควบคุมการหมุนของเซอร์โวมอเตอร์ทั้ง 4 ตัวให้ทำงานสัมพันธ์กับโครงสร้างของแขนกลที่ประกอบขึ้น โดยจะเริ่มทำการทดสอบทีละส่วนเพื่อให้ง่ายต่อการเรียนรู้และปรับแก้โปรแกรม การเขียนโปรแกรมเริ่มจากการควบคุมส่วนฐาน หัวไหล่ ข้อศอก มือจับ และให้หุ่นยนต์ทุกส่วนทำงานแบบอัตโนมัติ ดังตัวอย่างที่ 10.1-10.6

ตัวอย่าง 10.1 โปรแกรมการควบคุมตำแหน่งฐานของหุ่นยนต์ให้หมุน 0, 90 และ 180 องศา

```
#include <Servo.h>
Servo Base;
int LEFT=180,CENTER=90,RIGHT=0;
void setup()
{ Base.attach(9);
  delay(2000);
}
void loop()
 {
   Base.write(RIGHT);
   delay(2000);
   Base.write(CENTER);
   delay(2000);
   Base.write(LEFT);
   delay(4000);
 }
```



รูปที่ 10.10 เซอร์โวมอเตอร์หมุน 0, 90 และ 180 องศา

ผลการรันโปรแกรม

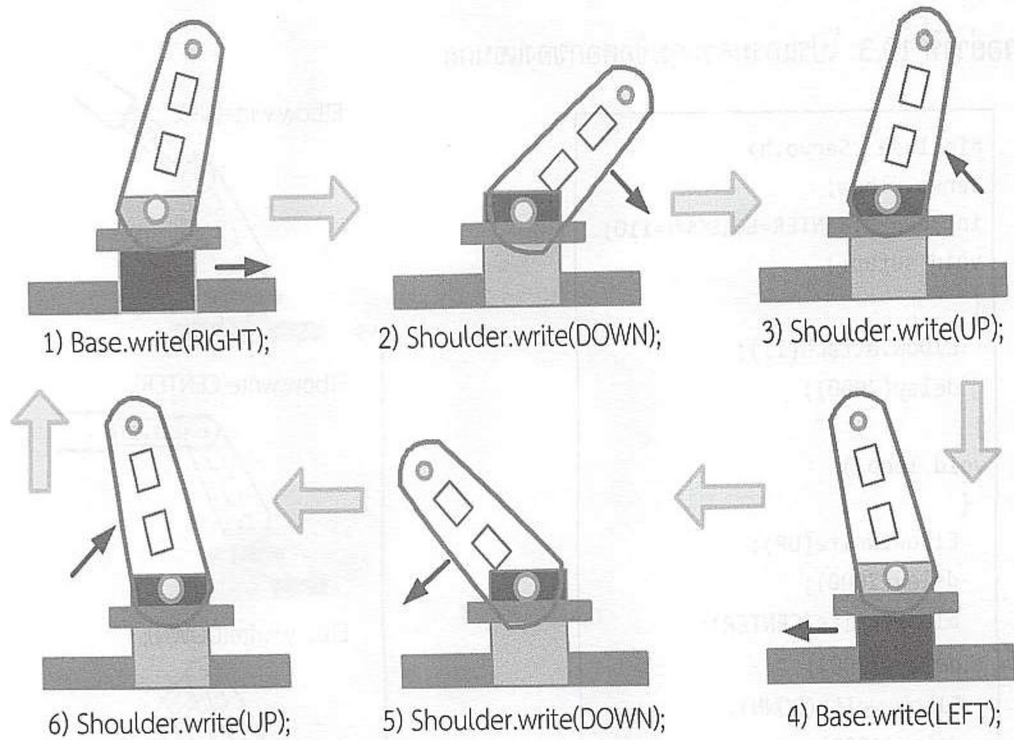
ไมโครคอนโทรลเลอร์จะควบคุมให้ฐานของแขนกลหรือเซอร์โวมอเตอร์ตัวที่ 1 หมุนฐานไปทางขวา หมุนมาตรงกลาง และหมุนไปทางซ้าย แล้ววนกลับไปทำงานซ้ำเดิม

ตัวอย่างที่ 10.2 โปรแกรมควบคุมฐานและหัวไหล่ของแขนกล

```
#include <Servo.h>                          // เรียกใช้ Servo.h
Servo Base;
Servo Shoulder;
int UP=90,DOWN=150,LEFT=60,RIGHT=140;       // ประกาศตัวแปรและกำหนด
                                            //   ค่าเริ่มต้น
void setup()
 { Base.attach(9);                          // ขา 9 ต่อกับ Servo Base
   Shoulder.attach(10);                     // ขา 10 ต่อกับ Servo Shoulder
   // Set Home
   delay(1000);
   Base.write(90); delay(3000);             // หมุนฐานไปที่ตำแหน่ง 90 องศา
   Shoulder.write(100); delay(1000);        // หมุนหัวไหล่ไปที่ตำแหน่ง 100 องศา
 }
void loop()
 { Base.write(RIGHT); delay(2000);          // หมุนฐานไปทางขวา
   Shoulder.write(DOWN); delay(2000);       // หมุนหัวไหล่ลง
   Shoulder.write(UP); delay(3000);         // ยกหัวไหล่ขึ้น
   Base.write(LEFT); delay(2000);           // หมุนฐานไปทางซ้าย
   Shoulder.write(DOWN); delay(2000);       // หมุนหัวไหล่ลง
   Shoulder.write(UP); delay(3000);         // ยกหัวไหล่ขึ้น
 }
```

ผลการรันโปรแกรม

ไมโครคอนโทรลเลอร์จะควบคุมให้แขนกลหมุนฐานไปทางขวา หมุนหัวไหล่ลง ยกหัวไหล่ขึ้น หมุนฐานไปทางซ้าย หมุนหัวไหล่ลง และยกหัวไหล่ขึ้น แล้ววนกลับไปทำงานซ้ำเดิมดังรูปที่ 10.11

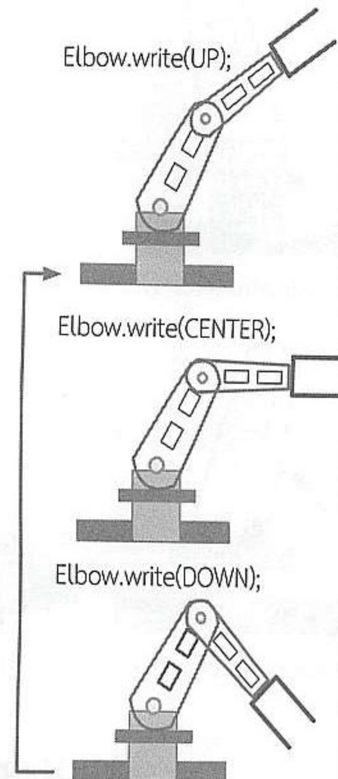


รูปที่ 10.11 การทำงานของแขนกลในตัวอย่างที่ 10.2

ในการควบคุมการทำงาน หากตำแหน่งของฐานและหัวไหล่ไม่เป็นไปตามที่กำหนด ต้องเปลี่ยนค่าในตัวแปร int UP = 90, DOWN = 150, LEFT = 60, RIGHT = 140 เช่น กำหนด LEFT= 60 แล้วฐานแขนกลยังหมุนมาทางซ้ายไม่ตรงตำแหน่งที่ต้องการก็ต้องปรับเป็น 50 หรือ 40 เนื่องจากการประกอบเซอร์โวมอเตอร์ ตำแหน่งของเฟืองไม่ตรงกันดังนั้นต้องปรับค่าให้เหมาะสม

ตัวอย่างที่ 10.3 โปรแกรมควบคุมข้อศอกของแขนกล

```
#include <Servo.h>
Servo Elbow;
int UP=50,CENTER=80,DOWN=110;
void setup()
{
  Elbow.attach(11);
  delay(2000);
}
void loop()
 {
  Elbow.write(UP);
  delay(2000);
  Elbow.write(CENTER);
  delay(2000);
  Elbow.write(DOWN);
  delay(4000);
 }
```



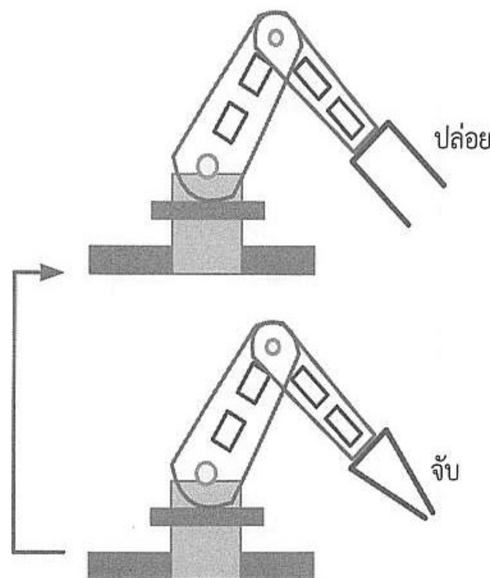
รูปที่ 10.12 การทำงานของข้อศอกแขนกล

ผลการรันโปรแกรม

ไมโครคอนโทรลเลอร์จะควบคุมเซอร์โวมอเตอร์ตัวที่ 3 ให้ทำหน้าที่ยกข้อศอกขึ้น หมุนข้อศอกมาตรงกลาง และหมุนข้อศอกลง แล้ววนกลับไปทำงานซ้ำเดิมดังรูป

ตัวอย่างที่ 10.4 โปรแกรมควบคุมมือจับของแขนกล

```
#include <Servo.h>
Servo Gripper;
int OPEN=50,CLOSE=115;
void setup()
{
  Gripper.attach(12);
  delay(2000);
}
void loop()
 {
   Gripper.write(OPEN);
   delay(2000);
   Gripper.write(CLOSE);
   delay(4000);
 }
```



รูปที่ 10.13 การปล่อยและจับของแขนกล

ผลการรันโปรแกรม

ไมโครคอนโทรลเลอร์จะควบคุมให้มือจับของแขนกลหรือเซอร์โวมอเตอร์ตัวที่ 4 ทำการหยิบจับและปล่อยชิ้นงาน แล้ววนกลับไปทำงานซ้ำเดิมดังรูป

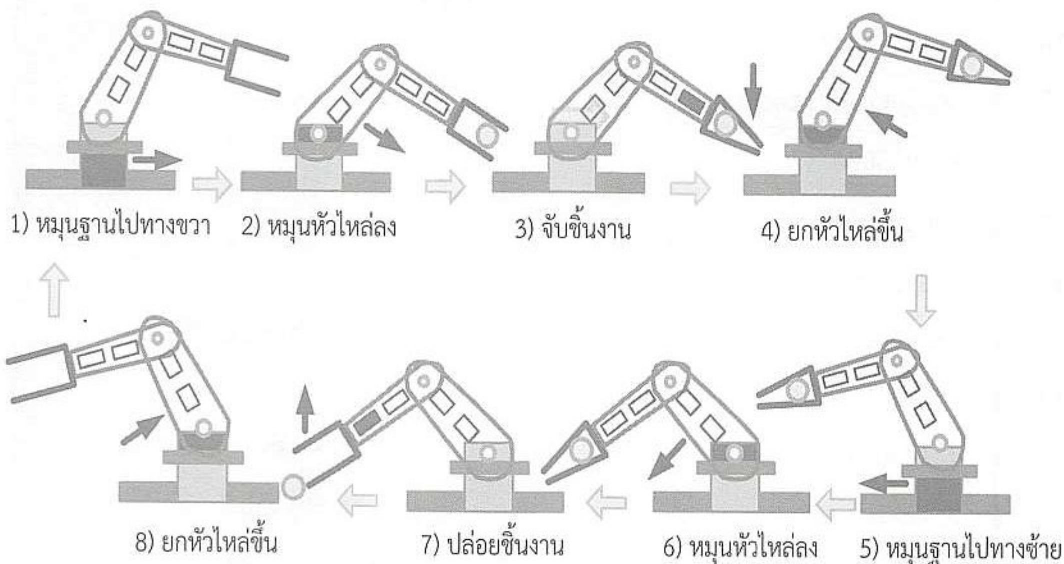
ตัวอย่างที่ 10.5 โปรแกรมควบคุมแขนกลให้ย้ายสิ่งของอัตโนมัติ

```
#include <Servo.h>                              // เรียกใช้ Servo.h
Servo Base;
Servo Shoulder;
Servo Elbow;
Servo Gripper;
int CLOSE=115,OPEN=50;                          // กำหนดค่าการจับและปล่อย
int UP=140,DOWN=105,LEFT=60,RIGHT=140;          // กำหนดค่าการหมุนขึ้น ลง ซ้าย ขวา
void setup()                                    // กำหนดขาสัญญาณ
{ Base.attach(9);                               // Servo 1 ต่อขา 9
  Shoulder.attach(10);                          // Servo 2 ต่อขา 10
  Elbow.attach(11);                             // Servo 3 ต่อขา 11
  Gripper.attach(12);                           // Servo 4 ต่อขา 12
  // Set Home                                   // กำหนดตำแหน่งเริ่มต้นของแขนกล
  delay(3000);
  Base.write(RIGHT);                            // หมุนฐานไปทางขวา
  delay(3000);
  Shoulder.write(UP);                           // ยกหัวไหล่ขึ้น
  delay(3000);
  Elbow.write(100);                             // หมุนข้อศอกไปที่ตำแหน่ง 100 องศา
  delay(2000);
  Gripper.write(OPEN);                          // เปิดมือจับเพื่อหยิบชิ้นงาน
}
void loop()                                     // วนรอบตลอดกาล
{                                               // เริ่มการทำงาน
  Base.write(RIGHT);                            // หมุนฐานไปทางขวา
  delay(2000);
```

```
  Shoulder.write(DOWN);         // หมุนหัวไหล่ลง
  delay(2000);
  Gripper.write(CLOSE);         // จับชิ้นงาน
  delay(2000);
  Shoulder.write(UP);           // ยกหัวไหล่ขึ้น
  delay(3000);
  Base.write(LEFT);             // หมุนฐานไปทางซ้าย
  delay(2000);
  Shoulder.write(DOWN);         // หมุนหัวไหล่ลง
  delay(2000);
  Gripper.write(OPEN);          // ปล่อยชิ้นงาน
  delay(3000);
  Shoulder.write(UP);           // ยกหัวไหล่ขึ้น
  delay(5000);                  // หน่วงเวลา 5 วินาที
}                               // จบการทำงาน
```

ผลการรันโปรแกรม

ไมโครคอนโทรลเลอร์จะควบคุมให้แขนกลทำงานตามลำดับ 1-8 ดังนี้ 1) หมุนฐานไปทางขวา 2) หมุนหัวไหล่ลง 3) จับชิ้นงาน 4) ยกหัวไหล่ขึ้น 5) หมุนฐานไปทางซ้าย 6) หมุนหัวไหล่ลง 7) ปล่อยชิ้นงาน 8) ยกหัวไหล่ขึ้น แล้ววนกลับไปทำงานซ้ำเดิมดังรูป



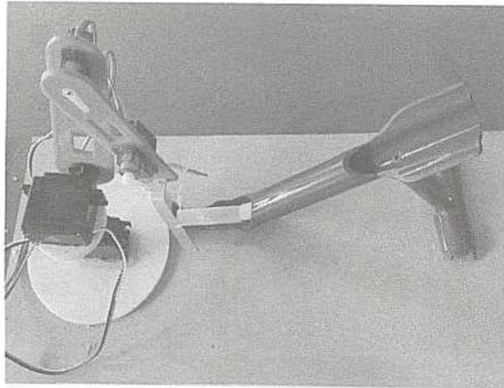
รูปที่ 10.14 การทำงานของแขนกลในตัวอย่างที่ 10.5

ตัวอย่างที่ 10.6 โปรแกรมควบคุมแขนกลให้จับสิ่งของมาวางบนราง

```
#include <Servo.h>                 // เรียกใช้ Servo.h
Servo Base;
Servo Shoulder;
Servo Elbow;
Servo Gripper;
int OPEN=50,CLOSE=120;
void setup()                       // การกำหนดขาสัญญาณและค่าเริ่มต้น
{
   Base.attach(9);                 // Servo 1 ต่อกับขา 9
   Shoulder.attach(10);            // Servo 2 ต่อกับขา 10
   Elbow.attach(11);               // Servo 3 ต่อกับขา 11
   Gripper.attach(12);             // Servo 4 ต่อกับขา 12
   // Set Home
   delay(1000);                    // หน่วงเวลา 1 วินาที
   Base.write(90);                 // หมุนฐานไปที่ตำแหน่ง 90 องศา
   delay(1000);
   Elbow.write(35);                // หมุนข้อศอกไปที่ตำแหน่ง 35 องศา
   delay(1000);
   Gripper.write(OPEN);            // เปิดมือจับ
   delay(3000);
 }
 void loop()
   {
   Shoulder.write(52);             // หมุนหัวไหล่ไปที่ตำแหน่ง 52 องศา
   delay(1000) ;
   Gripper.write(CLOSE);           // จับสิ่งของ
   delay(1000);
   Shoulder.write(60);             // หมุนหัวไหล่ไปที่ตำแหน่ง 60 องศา
   delay(500);
   Elbow.write(140);               // หมุนข้อศอกไปที่ตำแหน่ง 140 องศา
   delay(1000);
   Shoulder.write(30);             // หมุนหัวไหล่ไปที่ตำแหน่ง 30 องศา
   delay(2000);
   Shoulder.write(20);             // หมุนหัวไหล่ไปที่ตำแหน่ง 20 องศา
   delay(1000);
   Gripper.write(OPEN);            // ปล่อยสิ่งของลงบนราง
   delay(1000);
   Shoulder.write(60);             // หมุนหัวไหล่ไปที่ตำแหน่ง 60 องศา
   delay(1000);
   Elbow.write(35);                // หมุนข้อศอกไปที่ตำแหน่ง 35 องศา
   delay(3000);
   }
```

ผลการรันโปรแกรม

ไมโครคอนโทรลเลอร์จะควบคุมให้แขนกลหมุนแขนลงเพื่อไปหยิบสิ่งของจากด้านล่าง แล้วยกหัวไหล่ขึ้น กางข้อศอกออก จากนั้นนำสิ่งของไปวางลงบนราง เสร็จแล้วจึงวนกลับไปทำงานซ้ำ



1) หมุนแขนลง



2) หยิบสิ่งของ



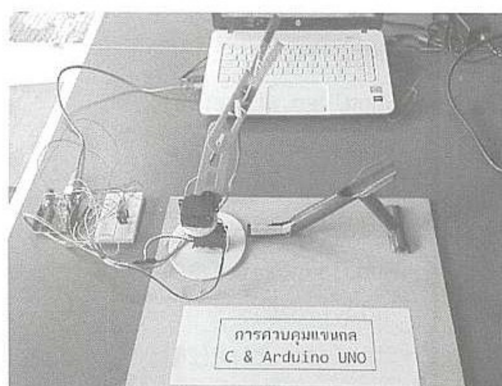
3) ยกแขนขึ้น



4) กางข้อศอกออก



5) ปล่อยสิ่งของ



6) ยกแขนขึ้นเพื่อเริ่มใหม่

รูปที่ 10.15 การทำงานของแขนกลในตัวอย่างที่ 10.6

10.8 สรุป

แขนกล คือหุ่นยนต์ที่มีฐานยึดติดกับที่ มักถูกนำมาใช้ในอุตสาหกรรมการผลิตเพื่อใช้แทนแรงงานมนุษย์ในงานที่มีลักษณะซ้ำ ๆ งานอันตราย งานหนัก งานที่น่าเบื่อ หรืองานที่ต้องการความเที่ยงตรงแม่นยำสูง แขนกลอุตสาหกรรมที่ใช้โดยทั่วไป ได้แก่ แขนกลในโรงงานประกอบรถยนต์ แขนกลสำหรับงานเชื่อมและงานประกอบ

โครงสร้างของแขนกลประกอบด้วย ฐานของหุ่นยนต์ ชิ้นส่วนที่เป็นแขนกลหรือส่วนเชื่อมต่อ (Link) ข้อต่อหรือจุดหมุน (Joint) และปลายของแขนกล เช่น มือจับ หัวเจาะ หัวเชื่อม หรือหัวพ่นสี

การแบ่งชนิดของแขนกลจะแบ่งตามลักษณะรูปทรงของพื้นที่ทำงาน ซึ่งสามารถแบ่งชนิดของแขนกลได้ดังต่อไปนี้

1) **แขนกลชนิด Cartesian** แขนกลชนิดนี้จะเคลื่อนที่เป็นเส้นตรงทั้ง 3 แกน มีข้อดีคือควบคุมตำแหน่งได้ง่ายเนื่องจากทุกแกนเคลื่อนที่เป็นเส้นตรง เช่น ลักษณะการทำงานของพล็อตเตอร์และเครื่องพิมพ์สามมิติ

2) **แขนกลชนิด Cylindrical** แขนกลชนิดนี้มีพื้นที่ทำงานเป็นรูปทรงกระบอก เคลื่อนที่แบบหมุนและเส้นตรงผสมกัน มีข้อดีคือการควบคุมตำแหน่งไม่ซับซ้อนและสามารถทำงานในพื้นที่แคบ ๆ หรือมีลักษณะเป็นโพรงได้

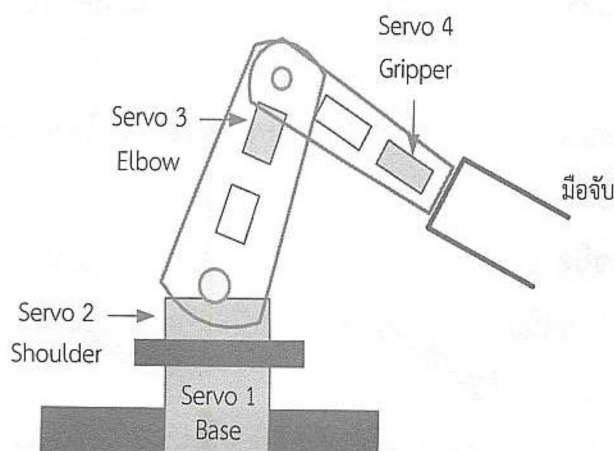
3) **แขนกลชนิด Spherical** แขนกลชนิดนี้มีพื้นที่ทำงานเป็นรูปทรงกลม เคลื่อนที่ในแนวเส้นตรงและหมุนผสมกัน การควบคุมตำแหน่งและการเคลื่อนที่มีความซับซ้อน มีข้อดีคือสามารถหมุนในมุมต่าง ๆ ได้

4) **แขนกลชนิด SCARA** แขนกลชนิดนี้การเคลื่อนที่ส่วนฐานและข้อศอกจะเป็นแบบหมุน ส่วนการเคลื่อนที่ในแนวตั้งจะเป็นการเคลื่อนที่ขึ้นลงแบบเส้นตรง มีข้อดีคือสามารถเคลื่อนที่ได้เร็วในแนวระนาบและมีความแม่นยำสูง

5) **แขนกลชนิด Articulated** แขนกลชนิดนี้ทุกแกนเคลื่อนที่แบบหมุน มีการเคลื่อนที่คล้ายกับแขนคน ซึ่งประกอบด้วยช่วงเอวหรือฐาน แขนท่อนบน แขนท่อนล่าง และมือจับ แต่เนื่องจากมีระบบพิกัดที่ซับซ้อนทำให้ควบคุมการเคลื่อนที่แบบเส้นตรงทำได้ยาก

คำถามท้ายบทที่ 10

1. หุ่นยนต์คืออะไร
2. แขนกลมีกี่ชนิด อะไรบ้าง
3. จากรูปจงอธิบายหน้าที่ของเซอร์โวมอเตอร์ 1-4



Servo 1 ทำหน้าที่ ....

Servo 2 ทำหน้าที่ ....

Servo 3 ทำหน้าที่ ....

Servo 4 ทำหน้าที่ ....

4. Link และ Joint ของแขนกลหมายถึงอะไร
5. จงอธิบายคำสั่งต่อไปนี้

   Base.attach(9);

   Base.write(0);

   Base.write(90);

   Base.write(180);

   delay(2000);
6. จงเขียนโปรแกรมควบคุมให้ฐานแขนกลหมุนซ้ายขวา 5 รอบ แล้วหยุดตามการกดสวิตซ์
7. จงเขียนโปรแกรมควบคุมให้แขนกลหมุนขนย้ายสิ่งของจากจุด A ไปจุด B 10 รอบ แล้วหยุดตามการกดสวิตซ์ ส่วนระยะการเคลื่อนที่และเงื่อนไขอื่น ๆ สามารถกำหนดเอง

บรรณานุกรม

1. ดอนสัน ปงผาบ. **ไมโครคอนโทรลเลอร์และการประยุกต์ใช้งาน 1.** กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2559.
2. ดอนสัน ปงผาบ. **ไมโครคอนโทรลเลอร์และการประยุกต์ใช้งาน 2.** กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2551.
3. Banzi, Massimo. **Getting Started with Arduino.** USA: O'Reilly Media, 2011.
4. Purdum, Jack. **Beginning C for Arduino.** USA: Apress, 2012.
5. โครงการสารานุกรมไทยสำหรับเยาวชนฯ. 2554. "ความหมาย ประวัติ และวิวัฒนาการของหุ่นยนต์." [ระบบออนไลน์]. แหล่งที่มา http://kanchanapisek.or.th/kp6/sub/book/book.php?book=36&chap=6&page=t36-6-infodetail01.html (12 กุมภาพันธ์ 2560).
6. วิกิพีเดีย. "หุ่นยนต์." [ระบบออนไลน์]. แหล่งที่มา http://th.wikipedia.org/wiki/หุ่นยนต์ (24 มกราคม 2560).
7. "Arduino." [ระบบออนไลน์]. แหล่งที่มา https://www.arduino.cc/en/Reference/HomePage (10 ตุลาคม 2559).
8. "C Programming Examples." [ระบบออนไลน์]. แหล่งที่มา http://www.programiz.com/c-programming/examples (31 กรกฎาคม 2559).
9. "Machine Specifications." [ระบบออนไลน์]. แหล่งที่มา http://www.toshiba-machine.com/csproductcatalog.aspx?dept=4 (24 มกราคม 2560).
10. "Robot Arm Tutorial." [ระบบออนไลน์]. แหล่งที่มา http://www.societyofrobots.com/robot_arm_tutorial.shtml (24 มกราคม 2560).

ประวัติผู้เขียน



ผู้ช่วยศาสตราจารย์ดอนสัน ปงผาบ
คณะเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยราชภัฏลำปาง

ประวัติการศึกษา

- คอ.บ. อิเล็กทรอนิกส์และคอมพิวเตอร์ (เกียรตินิยม) สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
- วศ.ม. วิศวกรรมไฟฟ้า สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ผลงานเขียนหนังสือ

1. "การเขียนโปรแกรมภาษาซีในงานควบคุม" สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น) และได้รับรางวัลหนังสือยอดนิยมของ ส.ส.ท.
2. "ไมโครคอนโทรลเลอร์และการประยุกต์ใช้งาน 1" สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
3. "ไมโครคอนโทรลเลอร์และการประยุกต์ใช้งาน 2" สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
4. "ไมโครคอนโทรลเลอร์ PIC และการประยุกต์ใช้งาน" สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)

แนะนำหนังสือ



ไมโครคอนโทรลเลอร์และการประยุกต์ใช้งาน 1 ฉบับปรับปรุง

(ฟรี!! CD สื่อประกอบการสอน)

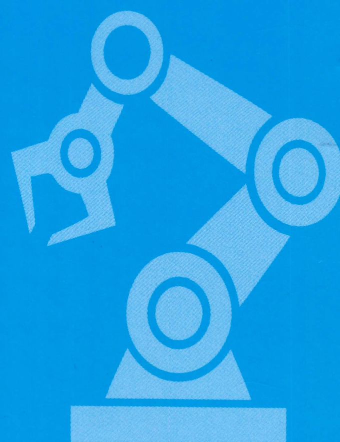
โดย ดอนสัน ปงผาบ

320 หน้า / ราคา 250 บาท

เนื้อหาประกอบด้วย ความรู้พื้นฐานของไมโครคอนโทรลเลอร์และไมโครโปรเซสเซอร์ โครงสร้างและสถาปัตยกรรมของไมโครคอนโทรลเลอร์ในตระกูล MCS-51 การเขียนโปรแกรมภาษาแอสเซมบลี พื้นฐานภาษาซี การใช้งานโปรแกรม Keil การเขียนโปรแกรมควบคุมอุปกรณ์อินพุตและเอาต์พุต การตรวจสอบ การจำลองการทำงานของโปรแกรม การพัฒนาไมโคร-คอนโทรลเลอร์ด้วยภาษาซีและหุ่นยนต์ขนาดเล็กควบคุมด้วยไมโครคอนโทรลเลอร์ การเขียนโปรแกรมควบคุมจอแสดงผล LCD อินเทอร์รัปต์ วงจรนับ วงจรจับเวลา และการเขียนโปรแกรมควบคุมหุ่นยนต์ 6 ขา

ภาษาซีและ Arduino

ลำดับเนื้อหาเข้าใจง่าย อธิบายสรุปเป็นขั้นตอน
พร้อมภาพประกอบและตัวอย่างมากมายเพื่อความเข้าใจที่มากขึ้น



เนื้อหาประกอบด้วย

- พื้นฐานภาษาซี
- การเขียนโปรแกรม DEV-C++
- ไมโครคอนโทรลเลอร์ Arduino UNO
- ดิจิทัลและแอนะล็อกอินพุต-เอาต์พุต
- การควบคุมมอเตอร์ เซอร์โวมอเตอร์
- ระบบควบคุมแบบ PID
- พื้นฐานหุ่นยนต์และแขนกล






ISBN 978-974-443-715-0

295 บาท

หมวดคอมพิวเตอร์